Red Hat Enterprise Linux 4 Cluster Suite Overview

Red Hat Cluster Suite for Red Hat Enterprise Linux
Edition 1.0

Landmann

Red Hat Enterprise Linux 4 Cluster Suite Overview

Red Hat Cluster Suite for Red Hat Enterprise Linux
Edition 1.0

Landmann
rlandmann@redhat.com

Legal Notice

Copyright © 2009 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Red Hat Cluster Suite Overview provides an overview of Red Hat Cluster Suite for Red Hat Enterprise Linux 4.

Table of Contents

Introduction **4**
1. Document Conventions 4
1.1. Typographic Conventions 5
1.2. Pull-quote Conventions 6
1.3. Notes and Warnings 7
2. Feedback 7

Chapter 1. Red Hat Cluster Suite Overview **9**
1.1. Cluster Basics 9
1.2. Red Hat Cluster Suite Introduction 10
1.3. Cluster Infrastructure 11
1.3.1. Cluster Management 11
1.3.2. Lock Management 13
1.3.3. Fencing 13
1.3.4. Cluster Configuration System 17
1.4. High-availability Service Management 18
1.5. Red Hat GFS 20
1.5.1. Superior Performance and Scalability 21
1.5.2. Performance, Scalability, Moderate Price 22
1.5.3. Economy and Performance 23
1.6. Cluster Logical Volume Manager 23
1.7. Global Network Block Device 25
1.8. Linux Virtual Server 26
1.8.1. Two-Tier LVS Topology 28
1.8.2. Three-Tier LVS Topology 30
1.8.3. Routing Methods 30
1.8.3.1. NAT Routing 31
1.8.3.2. Direct Routing 32
1.8.4. Persistence and Firewall Marks 33
1.8.4.1. Persistence 33
1.8.4.2. Firewall Marks 33
1.9. Cluster Administration Tools 33
1.9.1. Conga 34
1.9.2. Cluster Administration GUI 36
1.9.2.1. Cluster Configuration Tool 36
1.9.2.2. Cluster Status Tool 38
1.9.3. Command Line Administration Tools 39
1.10. Linux Virtual Server Administration GUI 39
1.10.1. CONTROL/MONITORING 40
1.10.2. GLOBAL SETTINGS 41
1.10.3. REDUNDANCY 43
1.10.4. VIRTUAL SERVERS 44
1.10.4.1. The VIRTUAL SERVER Subsection 44
1.10.4.2. REAL SERVER Subsection 46
1.10.4.3. EDIT MONITORING SCRIPTS Subsection 48

Chapter 2. Red Hat Cluster Suite Component Summary **50**
2.1. Cluster Components 50
2.2. Man Pages 55
2.3. Compatible Hardware 57

Revision History **58**

Index **58**
A 58
C 58
F 58
I 59
L 59
N 59
O 59
P 59
R 60
T 60

Introduction

This document provides a high-level overview of Red Hat Cluster Suite for Red Hat Enterprise Linux 4. Although the information in this document is an overview, you should have advanced working knowledge of Red Hat Enterprise Linux and understand the concepts of server computing to gain a good comprehension of the information. For more information about using Red Hat Enterprise Linux, refer to the following resources:

- *Red Hat Enterprise Linux Installation Guide* — Provides information regarding installation.
- *Red Hat Enterprise Linux Introduction to System Administration* — Provides introductory information for new Red Hat Enterprise Linux system administrators.
- *Red Hat Enterprise Linux System Administration Guide* — Provides more detailed information about configuring Red Hat Enterprise Linux to suit your particular needs as a user.
- *Red Hat Enterprise Linux Reference Guide* — Provides detailed information suited for more experienced users to reference when needed, as opposed to step-by-step instructions.
- *Red Hat Enterprise Linux Security Guide* — Details the planning and the tools involved in creating a secured computing environment for the data center, workplace, and home.

This document contains overview information about Red Hat Cluster Suite for Red Hat Enterprise Linux 4 and is part of a documentation set that provides conceptual, procedural, and reference information about Red Hat Cluster Suite for Red Hat Enterprise Linux 4.

Red Hat Cluster Suite documentation and other Red Hat documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Linux Documentation CD and online at http://www.redhat.com/docs/.

For more information about Red Hat Cluster Suite for Red Hat Enterprise Linux 4, refer to the following resources:

- *Configuring and Managing a Red Hat Cluster* — Provides information about installing, configuring and managing Red Hat Cluster components.
- *LVM Administrator's Guide: Configuration and Administration* — Provides a description of the Logical Volume Manager (LVM), including information on running LVM in a clustered environment.
- *Global File System: Configuration and Administration* — Provides information about installing, configuring, and maintaining Red Hat GFS (Red Hat Global File System).
- *Using Device-Mapper Multipath* — Provides information about using the Device-Mapper Multipath feature of Red Hat Enterprise Linux 4.
- *Using GNBD with Global File System* — Provides an overview on using Global Network Block Device (GNBD) with Red Hat GFS.
- *Linux Virtual Server Administration* — Provides information on configuring high-performance systems and services with the Linux Virtual Server (LVS).
- *Red Hat Cluster Suite Release Notes* — Provides information about the current release of Red Hat Cluster Suite.

Red Hat Cluster Suite documentation and other Red Hat documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Linux Documentation CD and online at http://www.redhat.com/docs/.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **`Buttons`** tab, select the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **`Search`** field and click **`Next`**. The character you sought will be highlighted in the **`Character Table`**. Double-click this highlighted character to place it in the **`Text to copy`** field and then click the **`Copy`** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                  printk(KERN_INFO "%s: device hasn't been assigned before, "
                    "so cannot be deassigned\n", __func__);
                  r = -EINVAL;
                  goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Feedback

If you spot a typo, or if you have thought of a way to make this document better, we would love to hear from you. Please submit a report in Bugzilla (http://bugzilla.redhat.com/bugzilla/) against the component **rh-cs-en**.

Be sure to mention the document's identifier:

```
Cluster_Suite_Overview(EN)-4.8 (2009-04-24:T15:25)
```

By mentioning this document's identifier, we know exactly which version of the guide you have.

If you have a suggestion for improving the documentation, try to be as specific as possible. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Chapter 1. Red Hat Cluster Suite Overview

Clustered systems provide reliability, scalability, and availability to critical production services. Using Red Hat Cluster Suite, you can create a cluster to suit your needs for performance, high availability, load balancing, scalability, file sharing, and economy. This chapter provides an overview of Red Hat Cluster Suite components and functions, and consists of the following sections:

- Section 1.1, “Cluster Basics”
- Section 1.2, “Red Hat Cluster Suite Introduction”
- Section 1.3, “Cluster Infrastructure”
- Section 1.4, “High-availability Service Management”
- Section 1.5, “Red Hat GFS”
- Section 1.6, “Cluster Logical Volume Manager”
- Section 1.7, “Global Network Block Device”
- Section 1.8, “Linux Virtual Server”
- Section 1.9, “Cluster Administration Tools”
- Section 1.10, “Linux Virtual Server Administration GUI”

1.1. Cluster Basics

A cluster is two or more computers (called *nodes* or *members*) that work together to perform a task. There are four major types of clusters:

- Storage
- High availability
- Load balancing
- High performance

Storage clusters provide a consistent file system image across servers in a cluster, allowing the servers to simultaneously read and write to a single shared file system. A storage cluster simplifies storage administration by limiting the installation and patching of applications to one file system. Also, with a cluster-wide file system, a storage cluster eliminates the need for redundant copies of application data and simplifies backup and disaster recovery. Red Hat Cluster Suite provides storage clustering through Red Hat GFS.

High-availability clusters provide continuous availability of services by eliminating single points of failure and by failing over services from one cluster node to another in case a node becomes inoperative. Typically, services in a high-availability cluster read and write data (via read-write mounted file systems). Therefore, a high-availability cluster must maintain data integrity as one cluster node takes over control of a service from another cluster node. Node failures in a high-availability cluster are not visible from clients outside the cluster. (High-availability clusters are sometimes referred to as failover clusters.) Red Hat Cluster Suite provides high-availability clustering through its High-availability Service Management component.

Load-balancing clusters dispatch network service requests to multiple cluster nodes to balance the request load among the cluster nodes. Load balancing provides cost-effective scalability because you can match the number of nodes according to load requirements. If a node in a load-balancing cluster becomes inoperative, the load-balancing software detects the failure and redirects requests to other cluster nodes. Node failures in a load-balancing cluster are not visible from clients outside the cluster. Red Hat Cluster Suite provides load-balancing through LVS (Linux Virtual Server).

High-performance clusters use cluster nodes to perform concurrent calculations. A high-performance

cluster allows applications to work in parallel, therefore enhancing the performance of the applications. (High performance clusters are also referred to as computational clusters or grid computing.)

Note

The cluster types summarized in the preceding text reflect basic configurations; your needs might require a combination of the clusters described.

1.2. Red Hat Cluster Suite Introduction

Red Hat Cluster Suite (RHCS) is an integrated set of software components that can be deployed in a variety of configurations to suit your needs for performance, high-availability, load balancing, scalability, file sharing, and economy.

RHCS consists of the following major components (refer to Figure 1.1, “Red Hat Cluster Suite Introduction”):

- Cluster infrastructure — Provides fundamental functions for nodes to work together as a cluster: configuration-file management, membership management, lock management, and fencing.
- High-availability Service Management — Provides failover of services from one cluster node to another in case a node becomes inoperative.
- Cluster administration tools — Configuration and management tools for setting up, configuring, and managing a Red Hat cluster. The tools are for use with the Cluster Infrastructure components, the High-availability and Service Management components, and storage.
- Linux Virtual Server (LVS) — Routing software that provides IP-Load-balancing. LVS runs in a pair of redundant servers that distributes client requests evenly to real servers that are behind the LVS servers.

You can supplement Red Hat Cluster Suite with the following components, which are part of an optional package (and *not* part of Red Hat Cluster Suite):

- Red Hat GFS (Global File System) — Provides a cluster file system for use with Red Hat Cluster Suite. GFS allows multiple nodes to share storage at a block level as if the storage were connected locally to each cluster node.
- Cluster Logical Volume Manager (CLVM) — Provides volume management of cluster storage.

  Note

  When you create or modify a CLVM volume for a clustered environment, you must ensure that you are running the **`clvmd`** daemon. For further information, refer to Section 1.6, “Cluster Logical Volume Manager”.

- Global Network Block Device (GNBD) — An ancillary component of GFS that exports block-level storage to Ethernet. This is an economical way to make block-level storage available to Red Hat GFS.

For a lower level summary of Red Hat Cluster Suite components and optional software, refer to Chapter 2, *Red Hat Cluster Suite Component Summary*.

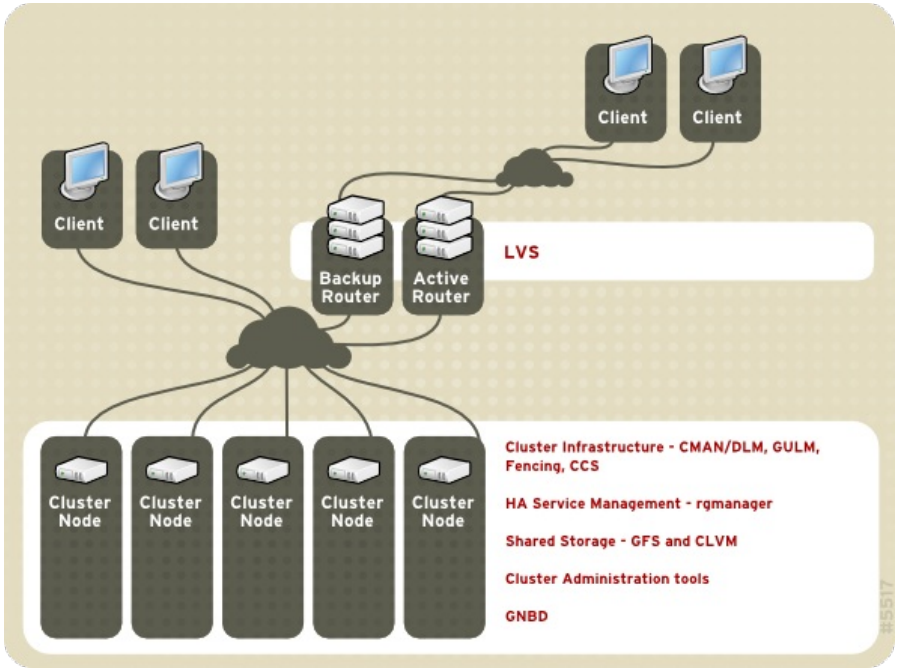

Figure 1.1. Red Hat Cluster Suite Introduction

Note

Figure 1.1, "Red Hat Cluster Suite Introduction" includes GFS, CLVM, and GNBD, which are components that are part of an optional package and *not* part of Red Hat Cluster Suite.

1.3. Cluster Infrastructure

The Red Hat Cluster Suite cluster infrastructure provides the basic functions for a group of computers (called *nodes* or *members*) to work together as a cluster. Once a cluster is formed using the cluster infrastructure, you can use other Red Hat Cluster Suite components to suit your clustering needs (for example, setting up a cluster for sharing files on a GFS file system or setting up service failover). The cluster infrastructure performs the following functions:

- Cluster management
- Lock management
- Fencing
- Cluster configuration management

1.3.1. Cluster Management

Cluster management manages cluster quorum and cluster membership. One of the following Red Hat Cluster Suite components performs cluster management: CMAN (an abbreviation for cluster manager) or GULM (Grand Unified Lock Manager). CMAN operates as the cluster manager if a cluster is configured to use DLM (Distributed Lock Manager) as the lock manager. GULM operates as the cluster manager if a cluster is configured to use GULM as the lock manager. The major difference between the two cluster managers is that CMAN is a distributed cluster manager and GULM is a client-server cluster manager. CMAN runs in each cluster node; cluster management is distributed across all nodes in the cluster (refer to Figure 1.2, "CMAN/DLM Overview"). GULM runs in nodes designated as GULM server nodes; cluster management is centralized in the nodes designated as GULM server nodes (refer to Figure 1.3, "GULM

Overview"). GULM server nodes manage the cluster through GULM clients in the cluster nodes. With GULM, cluster management operates in a limited number of nodes: either one, three, or five nodes configured as GULM servers.

The cluster manager keeps track of cluster quorum by monitoring the count of cluster nodes that run cluster manager. (In a CMAN cluster, all cluster nodes run cluster manager; in a GULM cluster only the GULM servers run cluster manager.) If more than half the nodes that run cluster manager are active, the cluster has quorum. If half the nodes that run cluster manager (or fewer) are active, the cluster does not have quorum, and all cluster activity is stopped. Cluster quorum prevents the occurrence of a "split-brain" condition — a condition where two instances of the same cluster are running. A split-brain condition would allow each cluster instance to access cluster resources without knowledge of the other cluster instance, resulting in corrupted cluster integrity.

In a CMAN cluster, quorum is determined by communication of heartbeats among cluster nodes via Ethernet. Optionally, quorum can be determined by a combination of communicating heartbeats via Ethernet *and* through a quorum disk. For quorum via Ethernet, quorum consists of 50 percent of the node votes plus 1. For quorum via quorum disk, quorum consists of user-specified conditions.

Note

In a CMAN cluster, by default each node has one quorum vote for establishing quorum. Optionally, you can configure each node to have more than one vote.

In a GULM cluster, the quorum consists of a majority of nodes designated as GULM servers according to the number of GULM servers configured:

- Configured with one GULM server — Quorum equals one GULM server.
- Configured with three GULM servers — Quorum equals two GULM servers.
- Configured with five GULM servers — Quorum equals three GULM servers.

The cluster manager keeps track of membership by monitoring heartbeat messages from other cluster nodes. When cluster membership changes, the cluster manager notifies the other infrastructure components, which then take appropriate action. For example, if node A joins a cluster and mounts a GFS file system that nodes B and C have already mounted, then an additional journal and lock management is required for node A to use that GFS file system. If a cluster node does not transmit a heartbeat message within a prescribed amount of time, the cluster manager removes the node from the cluster and communicates to other cluster infrastructure components that the node is not a member. Again, other cluster infrastructure components determine what actions to take upon notification that node is no longer a cluster member. For example, Fencing would fence the node that is no longer a member.

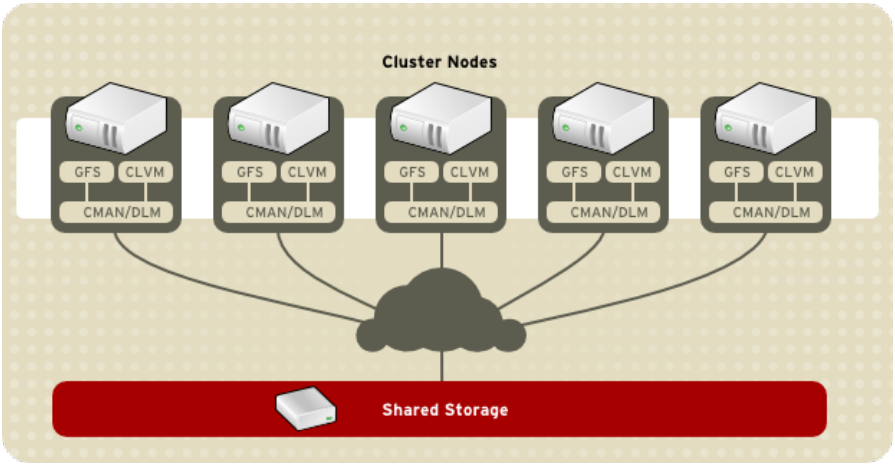

Figure 1.2. CMAN/DLM Overview

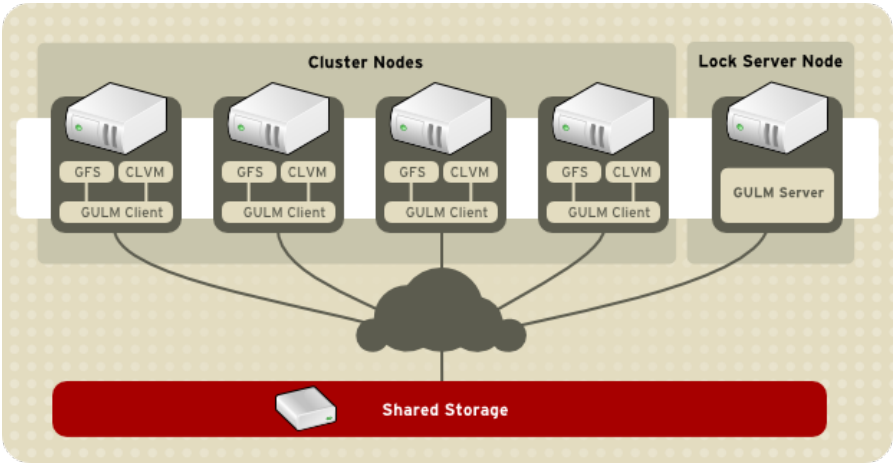

Figure 1.3. GULM Overview

1.3.2. Lock Management

Lock management is a common cluster-infrastructure service that provides a mechanism for other cluster infrastructure components to synchronize their access to shared resources. In a Red Hat cluster, one of the following Red Hat Cluster Suite components operates as the lock manager: DLM (Distributed Lock Manager) or GULM (Grand Unified Lock Manager). The major difference between the two lock managers is that DLM is a distributed lock manager and GULM is a client-server lock manager. DLM runs in each cluster node; lock management is distributed across all nodes in the cluster (refer to Figure 1.2, "CMAN/DLM Overview"). DLM can be the lock manager only in a cluster configured with CMAN as its cluster manager. GULM runs in nodes designated as GULM server nodes; lock management is centralized in the nodes designated as GULM server nodes. GULM server nodes manage locks through GULM clients in the cluster nodes (refer to Figure 1.3, "GULM Overview"). With GULM, lock management operates in a limited number of nodes: either one, three, or five nodes configured as GULM servers. GFS and CLVM use locks from the lock manager. GFS uses locks from the lock manager to synchronize access to file system metadata (on shared storage). CLVM uses locks from the lock manager to synchronize updates to LVM volumes and volume groups (also on shared storage).

1.3.3. Fencing

Fencing is the disconnection of a node from the cluster's shared storage. Fencing cuts off I/O from shared storage, thus ensuring data integrity.

The cluster infrastructure performs fencing through one of the following programs according to the type of cluster manager and lock manager that is configured:

- Configured with CMAN/DLM — **fenced**, the fence daemon, performs fencing.
- Configured with GULM servers — GULM performs fencing.

When the cluster manager determines that a node has failed, it communicates to other cluster-infrastructure components that the node has failed. The fencing program (either **fenced** or GULM), when notified of the failure, fences the failed node. Other cluster-infrastructure components determine what actions to take — that is, they perform any recovery that needs to done. For example, DLM and GFS (in a cluster configured with CMAN/DLM), when notified of a node failure, suspend activity until they detect that the fencing program has completed fencing the failed node. Upon confirmation that the failed

node is fenced, DLM and GFS perform recovery. DLM releases locks of the failed node; GFS recovers the journal of the failed node.

The fencing program determines from the cluster configuration file which fencing method to use. Two key elements in the cluster configuration file define a fencing method: fencing agent and fencing device. The fencing program makes a call to a fencing agent specified in the cluster configuration file. The fencing agent, in turn, fences the node via a fencing device. When fencing is complete, the fencing program notifies the cluster manager.

Red Hat Cluster Suite provides a variety of fencing methods:

- Power fencing — A fencing method that uses a power controller to power off an inoperable node
- Fibre Channel switch fencing — A fencing method that disables the Fibre Channel port that connects storage to an inoperable node
- GNBD fencing — A fencing method that disables an inoperable node's access to a GNBD server
- Other fencing — Several other fencing methods that disable I/O or power of an inoperable node, including IBM Bladecenters, PAP, DRAC/MC, HP ILO, IPMI, IBM RSA II, and others

Figure 1.4, “Power Fencing Example” shows an example of power fencing. In the example, the fencing program in node A causes the power controller to power off node D. Figure 1.5, “Fibre Channel Switch Fencing Example” shows an example of Fibre Channel switch fencing. In the example, the fencing program in node A causes the Fibre Channel switch to disable the port for node D, disconnecting node D from storage.

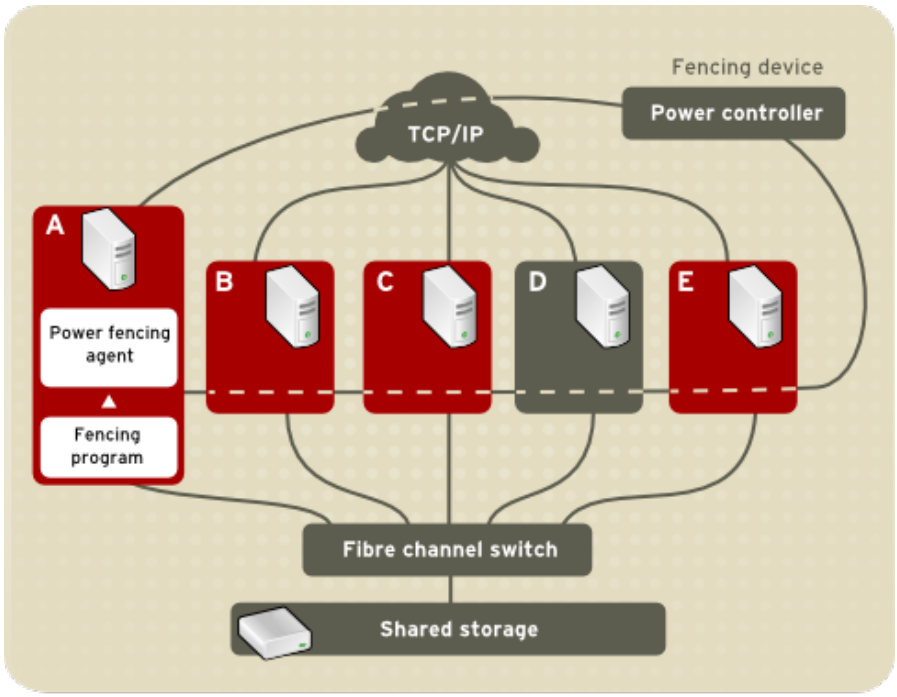

Figure 1.4. Power Fencing Example

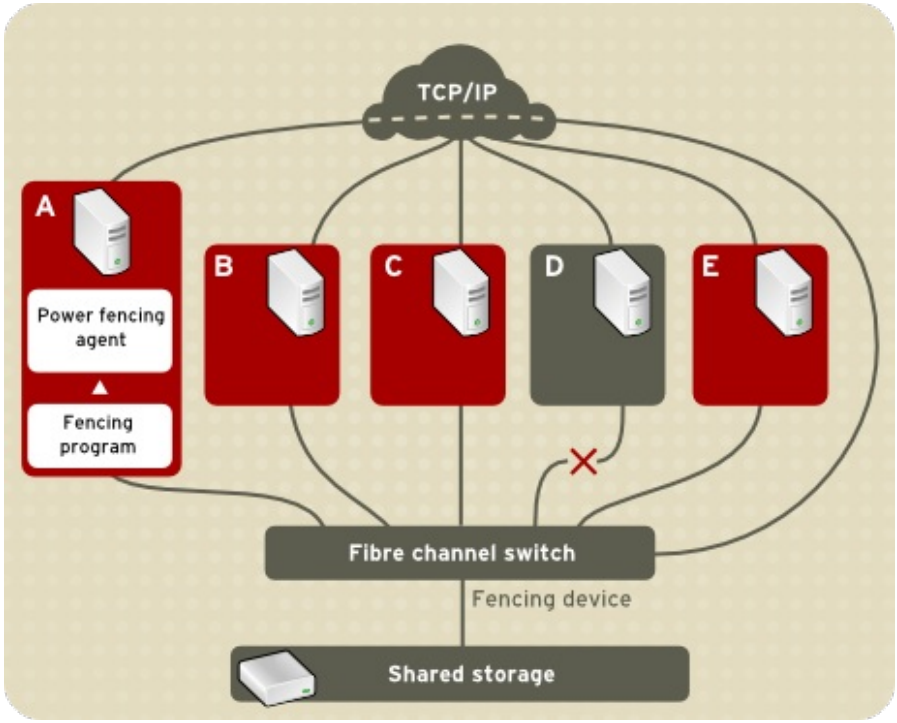

Figure 1.5. Fibre Channel Switch Fencing Example

Specifying a fencing method consists of editing a cluster configuration file to assign a fencing-method name, the fencing agent, and the fencing device for each node in the cluster.

Note

Other fencing parameters may be necessary depending on the type of cluster manager (either CMAN or GULM) selected in a cluster.

The way in which a fencing method is specified depends on if a node has either dual power supplies or multiple paths to storage. If a node has dual power supplies, then the fencing method for the node must specify at least two fencing devices — one fencing device for each power supply (refer to Figure 1.6, "Fencing a Node with Dual Power Supplies"). Similarly, if a node has multiple paths to Fibre Channel storage, then the fencing method for the node must specify one fencing device for each path to Fibre Channel storage. For example, if a node has two paths to Fibre Channel storage, the fencing method should specify two fencing devices — one for each path to Fibre Channel storage (refer to Figure 1.7, "Fencing a Node with Dual Fibre Channel Connections").

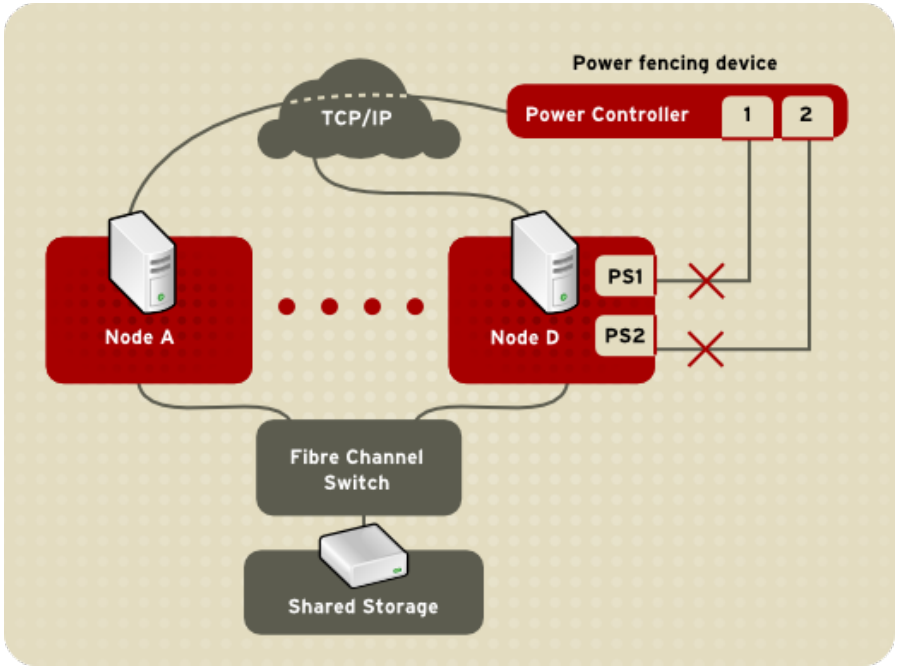

Figure 1.6. Fencing a Node with Dual Power Supplies

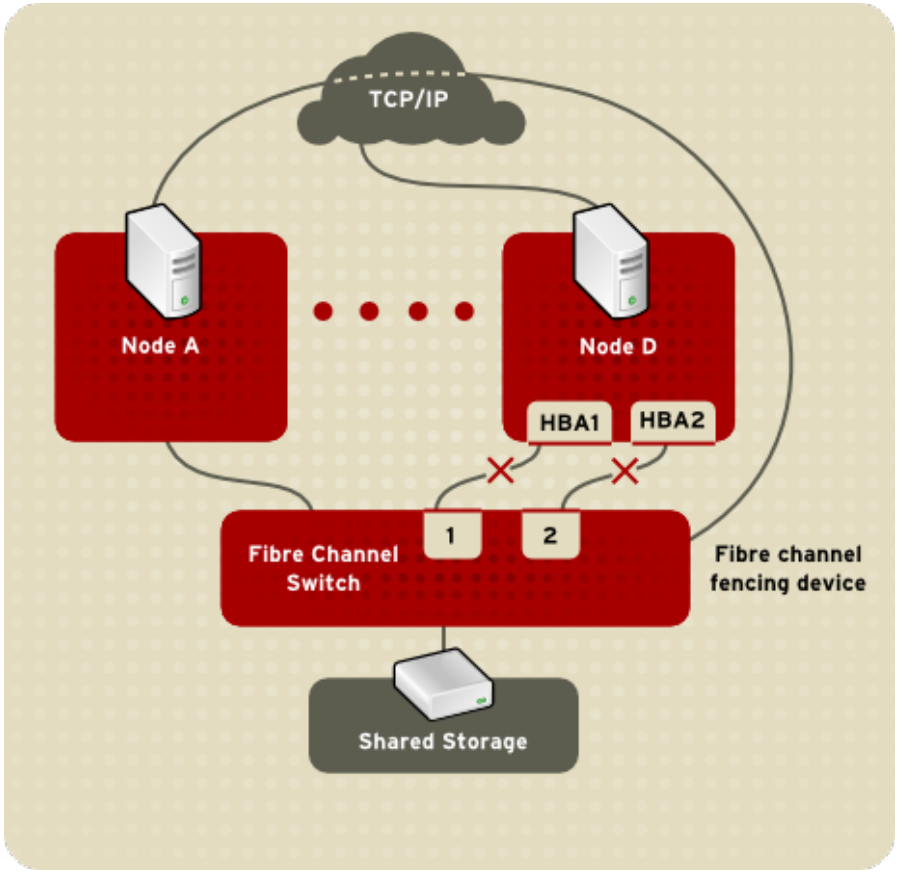

Figure 1.7. Fencing a Node with Dual Fibre Channel Connections

You can configure a node with one fencing method or multiple fencing methods. When you configure a node for one fencing method, that is the only fencing method available for fencing that node. When you configure a node for multiple fencing methods, the fencing methods are *cascaded* from one fencing method to another according to the order of the fencing methods specified in the cluster configuration

file. If a node fails, it is fenced using the first fencing method specified in the cluster configuration file for that node. If the first fencing method is not successful, the next fencing method specified for that node is used. If none of the fencing methods is successful, then fencing starts again with the first fencing method specified, and continues looping through the fencing methods in the order specified in the cluster configuration file until the node has been fenced.

1.3.4. Cluster Configuration System

The Cluster Configuration System (CCS) manages the cluster configuration and provides configuration information to other cluster components in a Red Hat cluster. CCS runs in each cluster node and makes sure that the cluster configuration file in each cluster node is up to date. For example, if a cluster system administrator updates the configuration file in Node A, CCS propagates the update from Node A to the other nodes in the cluster (refer to Figure 1.8, “CCS Overview”).

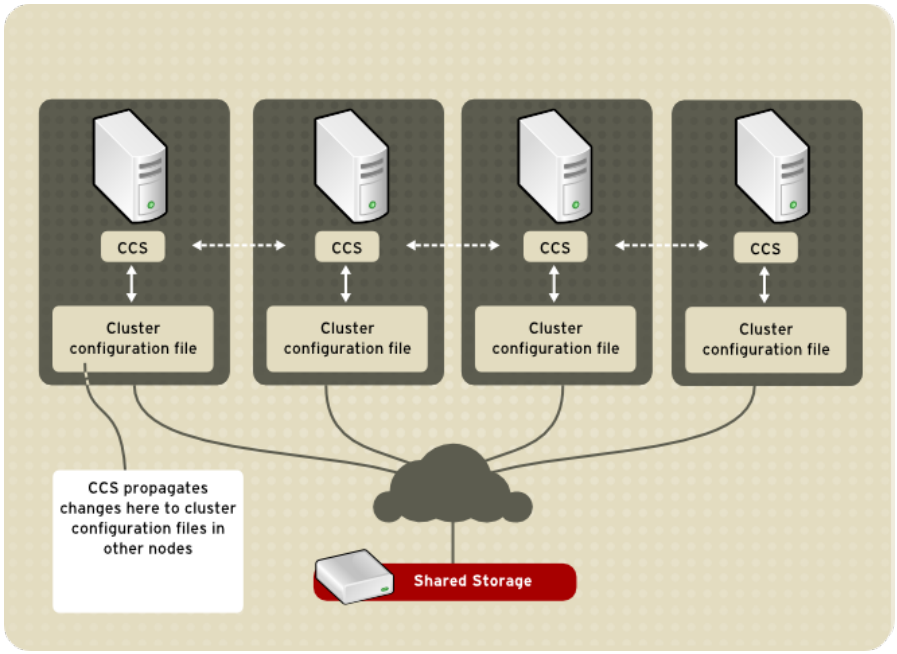

Figure 1.8. CCS Overview

Other cluster components (for example, CMAN) access configuration information from the configuration file through CCS (refer to Figure 1.8, “CCS Overview”).

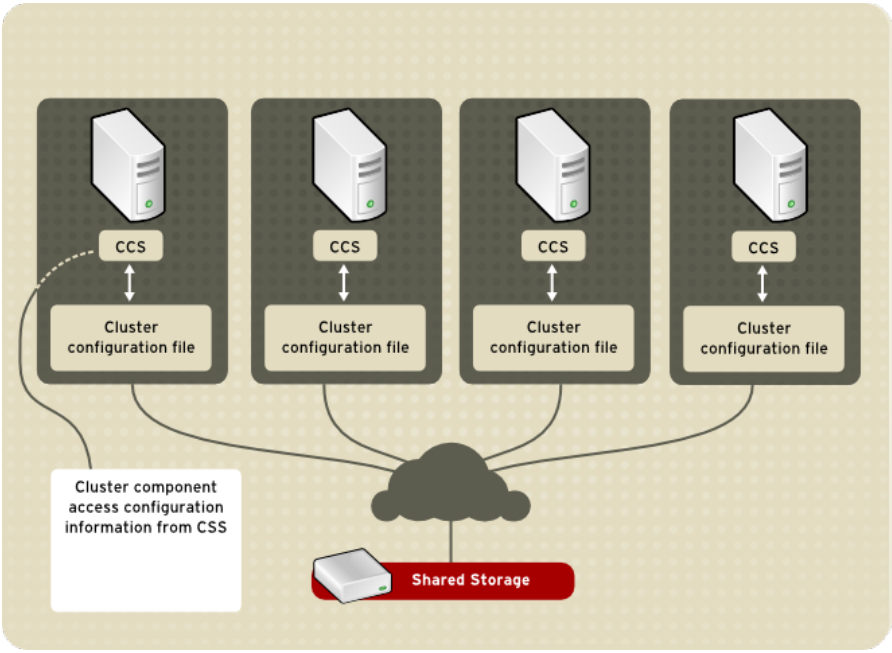

Figure 1.9. Accessing Configuration Information

The cluster configuration file (**/etc/cluster/cluster.conf**) is an XML file that describes the following cluster characteristics:

- Cluster name — Displays the cluster name, cluster configuration file revision level, locking type (either DLM or GULM), and basic fence timing properties used when a node joins a cluster or is fenced from the cluster.
- Cluster — Displays each node of the cluster, specifying node name, node ID, number of quorum votes, and fencing method for that node.
- Fence Device — Displays fence devices in the cluster. Parameters vary according to the type of fence device. For example for a power controller used as a fence device, the cluster configuration defines the name of the power controller, its IP address, login, and password.
- Managed Resources — Displays resources required to create cluster services. Managed resources includes the definition of failover domains, resources (for example an IP address), and services. Together the managed resources define cluster services and failover behavior of the cluster services.

1.4. High-availability Service Management

High-availability service management provides the ability to create and manage high-availability *cluster services* in a Red Hat cluster. The key component for high-availability service management in a Red Hat cluster, **rgmanager**, implements cold failover for off-the-shelf applications. In a Red Hat cluster, an application is configured with other cluster resources to form a high-availability cluster service. A high-availability cluster service can fail over from one cluster node to another with no apparent interruption to cluster clients. Cluster-service failover can occur if a cluster node fails or if a cluster system administrator moves the service from one cluster node to another (for example, for a planned outage of a cluster node).

To create a high-availability service, you must configure it in the cluster configuration file. A cluster service comprises cluster *resources*. Cluster resources are building blocks that you create and manage in the cluster configuration file — for example, an IP address, an application initialization script, or a Red Hat GFS shared partition.

You can associate a cluster service with a *failover domain*. A failover domain is a subset of cluster nodes that are eligible to run a particular cluster service (refer to Figure 1.10, “Failover Domains”).

Note

Failover domains are *not* required for operation.

A cluster service can run on only one cluster node at a time to maintain data integrity. You can specify failover priority in a failover domain. Specifying failover priority consists of assigning a priority level to each node in a failover domain. The priority level determines the failover order — determining which node that a cluster service should fail over to. If you do not specify failover priority, a cluster service can fail over to any node in its failover domain. Also, you can specify if a cluster service is restricted to run only on nodes of its associated failover domain. (When associated with an unrestricted failover domain, a cluster service can start on any cluster node in the event no member of the failover domain is available.)

In Figure 1.10, “Failover Domains”, Failover Domain 1 is configured to restrict failover within that domain; therefore, Cluster Service X can only fail over between Node A and Node B. Failover Domain 2 is also configured to restrict failover with its domain; additionally, it is configured for failover priority. Failover Domain 2 priority is configured with Node C as priority 1, Node B as priority 2, and Node D as priority 3. If Node C fails, Cluster Service Y fails over to Node B next. If it cannot fail over to Node B, it tries failing over to Node D. Failover Domain 3 is configured with no priority and no restrictions. If the node that Cluster Service Z is running on fails, Cluster Service Z tries failing over to one of the nodes in Failover Domain 3. However, if none of those nodes is available, Cluster Service Z can fail over to any node in the cluster.

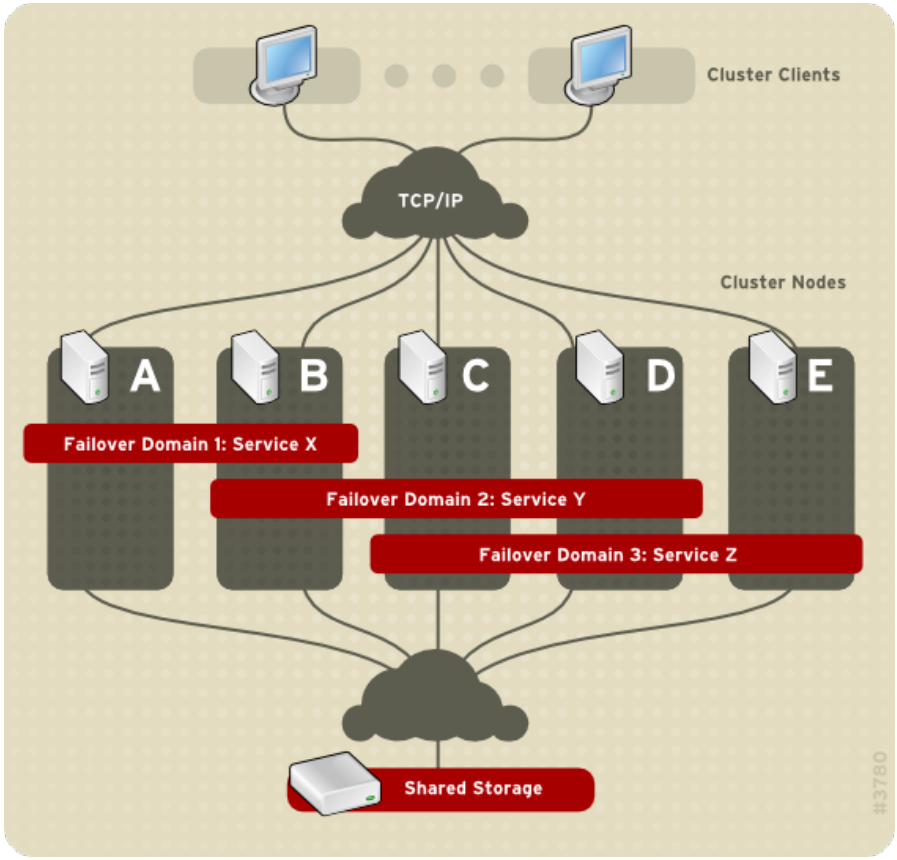

Figure 1.10. Failover Domains

Figure 1.11, “Web Server Cluster Service Example” shows an example of a high-availability cluster service that is a web server named "content-webserver". It is running in cluster node B and is in a failover domain that consists of nodes A, B, and D. In addition, the failover domain is configured with a failover priority to fail over to node D before node A and to restrict failover to nodes only in that failover domain. The cluster service comprises these cluster resources:

- IP address resource — IP address 10.10.10.201.
- An application resource named "httpd-content" — a web server application init script **`/etc/init.d/httpd`** (specifying **`httpd`**).
- A file system resource — Red Hat GFS named "gfs-content-webserver".

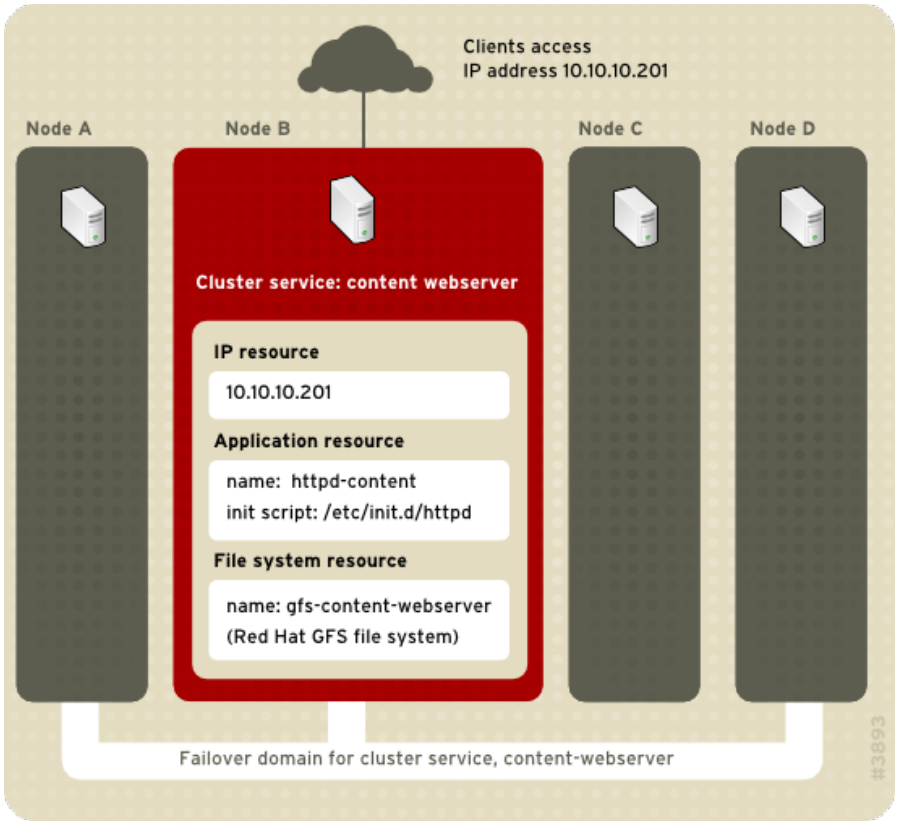

Figure 1.11. Web Server Cluster Service Example

Clients access the cluster service through the IP address 10.10.10.201, enabling interaction with the web server application, httpd-content. The httpd-content application uses the gfs-content-webserver file system. If node B were to fail, the content-webserver cluster service would fail over to node D. If node D were not available or also failed, the service would fail over to node A. Failover would occur with no apparent interruption to the cluster clients. The cluster service would be accessible from another cluster node via the same IP address as it was before failover.

1.5. Red Hat GFS

Red Hat GFS is a cluster file system that allows a cluster of nodes to simultaneously access a block device that is shared among the nodes. GFS is a native file system that interfaces directly with the VFS layer of the Linux kernel file-system interface. GFS employs distributed metadata and multiple journals for optimal operation in a cluster. To maintain file system integrity, GFS uses a lock manager to coordinate I/O. When one node changes data on a GFS file system, that change is immediately visible to

the other cluster nodes using that file system.

Using Red Hat GFS, you can achieve maximum application uptime through the following benefits:

- Simplifying your data infrastructure
  - Install and patch applications once for the entire cluster.
  - Eliminates the need for redundant copies of application data (duplication).
  - Enables concurrent read/write access to data by many clients.
  - Simplifies backup and disaster recovery (only one file system to back up or recover).
- Maximize the use of storage resources; minimize storage administration costs.
  - Manage storage as a whole instead of by partition.
  - Decrease overall storage needs by eliminating the need for data replications.
- Scale the cluster seamlessly by adding servers or storage on the fly.
  - No more partitioning storage through complicated techniques.
  - Add servers to the cluster on the fly by mounting them to the common file system.

Nodes that run Red Hat GFS are configured and managed with Red Hat Cluster Suite configuration and management tools. Volume management is managed through CLVM (Cluster Logical Volume Manager). Red Hat GFS provides data sharing among GFS nodes in a Red Hat cluster. GFS provides a single, consistent view of the file-system name space across the GFS nodes in a Red Hat cluster. GFS allows applications to install and run without much knowledge of the underlying storage infrastructure. Also, GFS provides features that are typically required in enterprise environments, such as quotas, multiple journals, and multipath support.

GFS provides a versatile method of networking storage according to the performance, scalability, and economic needs of your storage environment. This chapter provides some very basic, abbreviated information as background to help you understand GFS.

You can deploy GFS in a variety of configurations to suit your needs for performance, scalability, and economy. For superior performance and scalability, you can deploy GFS in a cluster that is connected directly to a SAN. For more economical needs, you can deploy GFS in a cluster that is connected to a LAN with servers that use *GNBD* (Global Network Block Device) or to *iSCSI* (Internet Small Computer System Interface) devices. (For more information about GNBD, refer to Section 1.7, “Global Network Block Device”.)

The following sections provide examples of how GFS can be deployed to suit your needs for performance, scalability, and economy:

- Section 1.5.1, “Superior Performance and Scalability”
- Section 1.5.2, “Performance, Scalability, Moderate Price”
- Section 1.5.3, “Economy and Performance”

Note

The GFS deployment examples reflect basic configurations; your needs might require a combination of configurations shown in the examples.

1.5.1. Superior Performance and Scalability

You can obtain the highest shared-file performance when applications access storage directly. The GFS SAN configuration in Figure 1.12, “GFS with a SAN” provides superior file performance for shared files

and file systems. Linux applications run directly on cluster nodes using GFS. Without file protocols or storage servers to slow data access, performance is similar to individual Linux servers with directly connected storage; yet, each GFS application node has equal access to all data files. GFS supports over 300 GFS nodes.

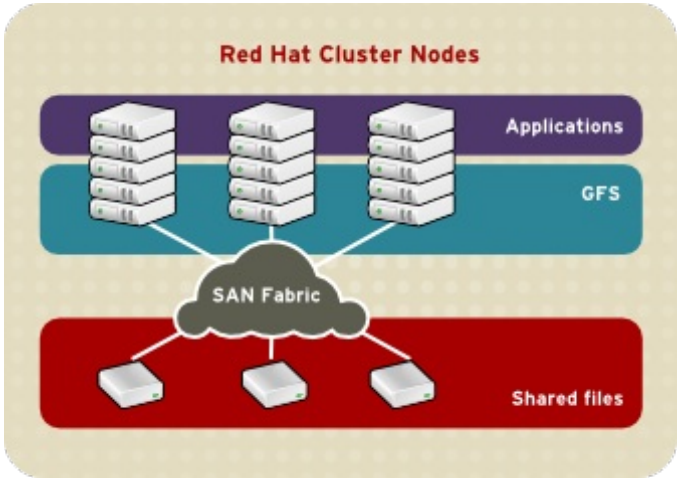

Figure 1.12. GFS with a SAN

1.5.2. Performance, Scalability, Moderate Price

Multiple Linux client applications on a LAN can share the same SAN-based data as shown in Figure 1.13, "GFS and GNBD with a SAN". SAN block storage is presented to network clients as block storage devices by GNBD servers. From the perspective of a client application, storage is accessed as if it were directly attached to the server in which the application is running. Stored data is actually on the SAN. Storage devices and data can be equally shared by network client applications. File locking and sharing functions are handled by GFS for each network client.

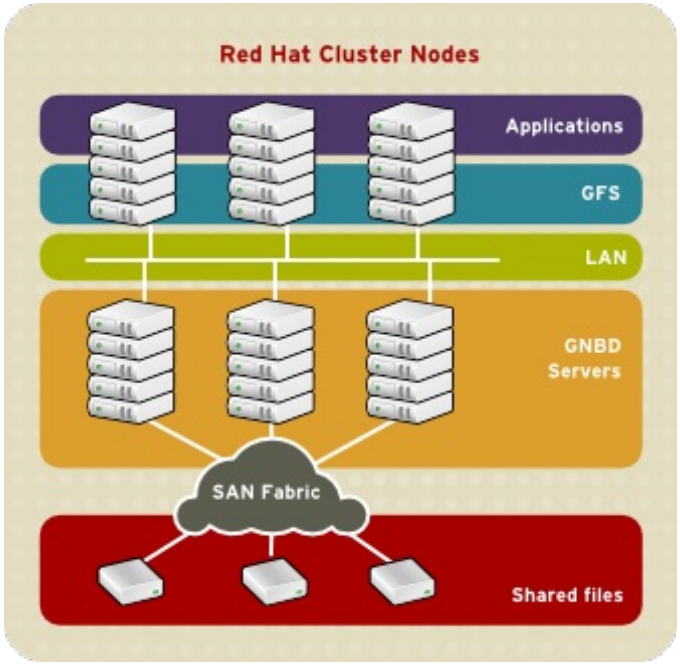

Figure 1.13. GFS and GNBD with a SAN

1.5.3. Economy and Performance

Figure 1.14, “GFS and GNBD with Directly Connected Storage” shows how Linux client applications can take advantage of an existing Ethernet topology to gain shared access to all block storage devices. Client data files and file systems can be shared with GFS on each client. Application failover can be fully automated with Red Hat Cluster Suite.

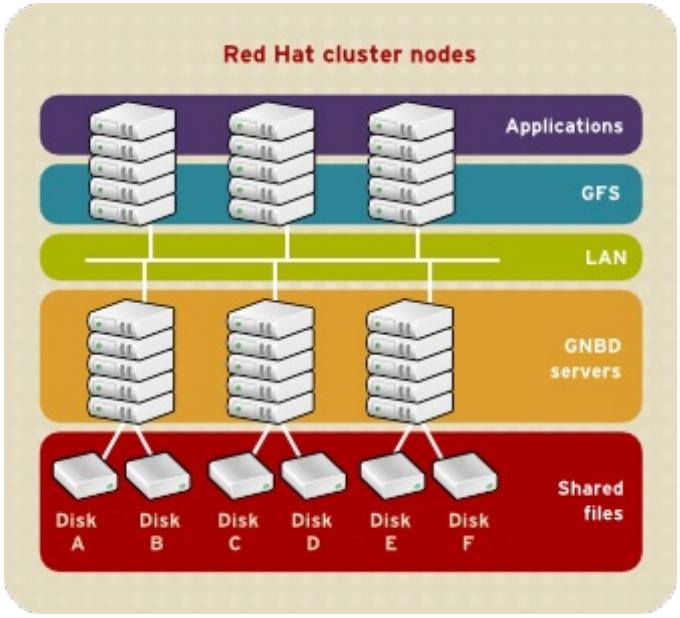

Figure 1.14. GFS and GNBD with Directly Connected Storage

1.6. Cluster Logical Volume Manager

The Cluster Logical Volume Manager (CLVM) provides a cluster-wide version of LVM2. CLVM provides the same capabilities as LVM2 on a single node, but makes the volumes available to all nodes in a Red Hat cluster. The logical volumes created with CLVM make logical volumes available to all nodes in a cluster.

The key component in CLVM is **clvmd**. **clvmd** is a daemon that provides clustering extensions to the standard LVM2 tool set and allows LVM2 commands to manage shared storage. **clvmd** runs in each cluster node and distributes LVM metadata updates in a cluster, thereby presenting each cluster node with the same view of the logical volumes (refer to Figure 1.15, “CLVM Overview”). Logical volumes created with CLVM on shared storage are visible to all nodes that have access to the shared storage. CLVM allows a user to configure logical volumes on shared storage by locking access to physical storage while a logical volume is being configured. CLVM uses the lock-management service provided by the cluster infrastructure (refer to Section 1.3, “Cluster Infrastructure”).

Note

Shared storage for use in Red Hat Cluster Suite requires that you be running the cluster logical volume manager daemon (**clvmd**) or the High Availability Logical Volume Management agents (HA-LVM). If you are not able to use either the **clvmd** daemon or HA-LVM for operational reasons or because you do not have the correct entitlements, you must not use single-instance LVM on the shared disk as this may result in data corruption. If you have any concerns please contact your Red Hat service representative.

Note

Using CLVM requires minor changes to **/etc/lvm/lvm.conf** for cluster-wide locking.

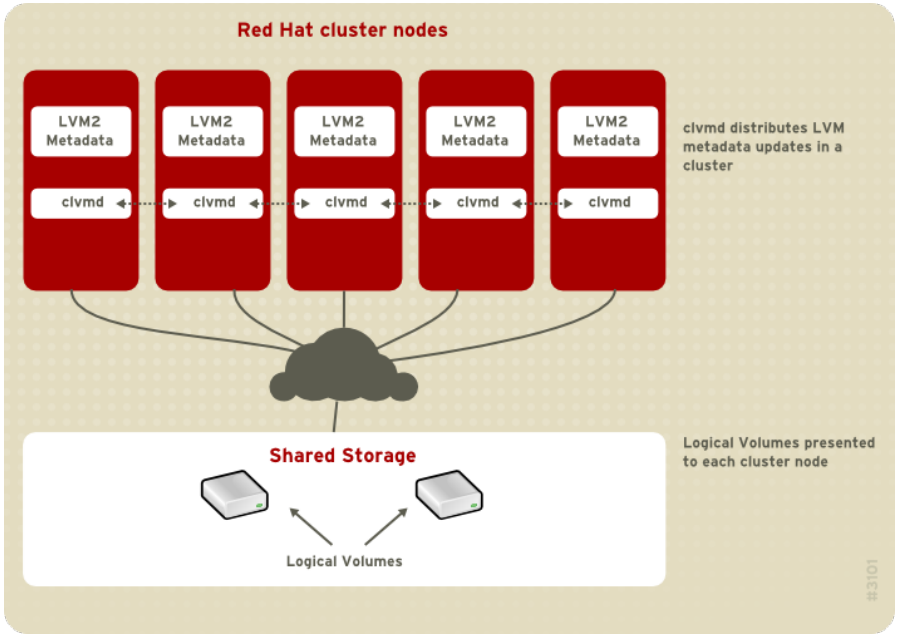

Figure 1.15. CLVM Overview

You can configure CLVM using the same commands as LVM2, using the LVM graphical user interface (refer to Figure 1.16, “LVM Graphical User Interface”), or using the storage configuration function of the **Conga** cluster configuration graphical user interface (refer to Figure 1.17, “Conga LVM Graphical User Interface”) . Figure 1.18, “Creating Logical Volumes” shows the basic concept of creating logical volumes from Linux partitions and shows the commands used to create logical volumes.

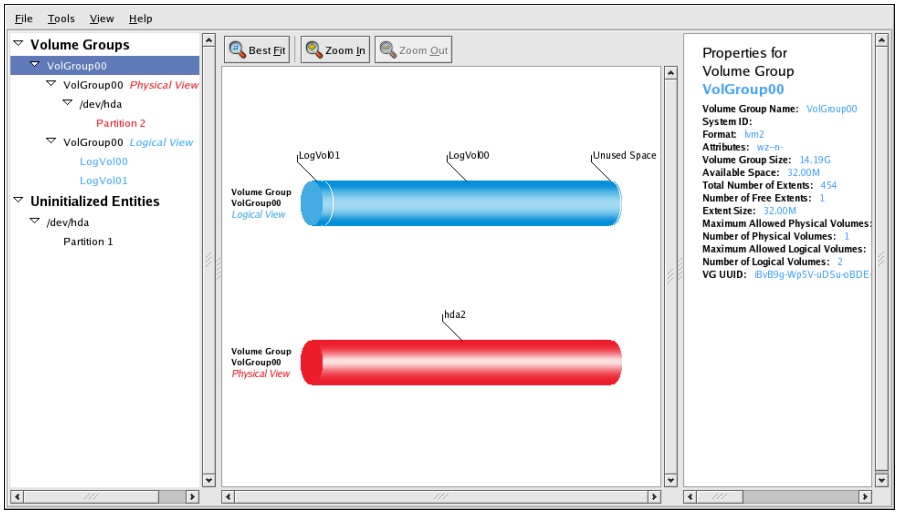

Figure 1.16. LVM Graphical User Interface

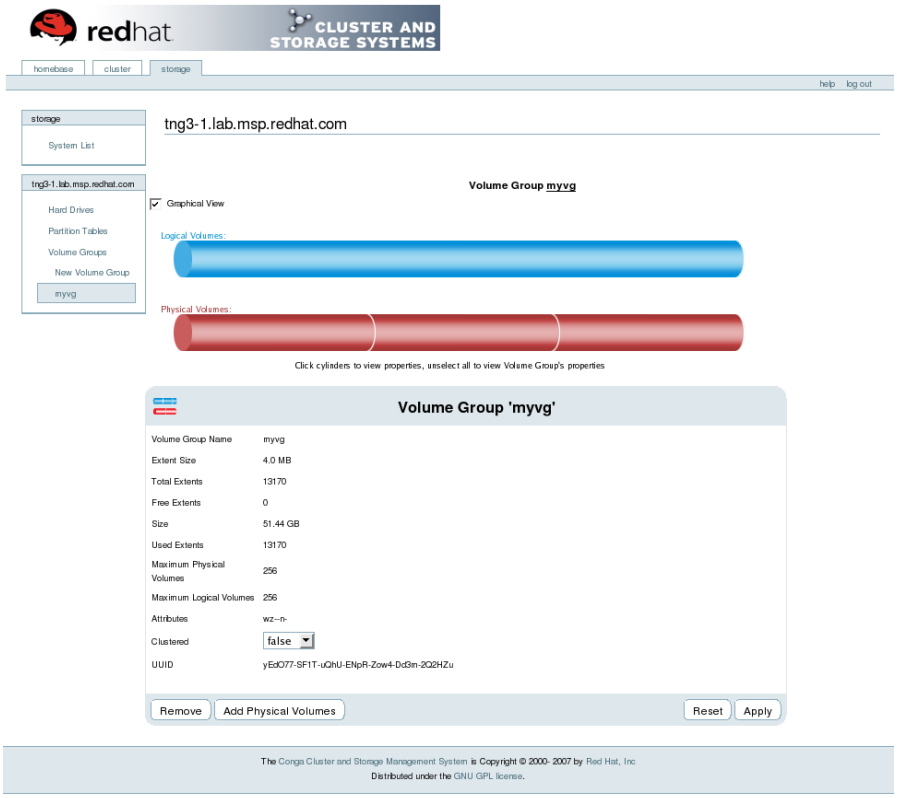

Figure 1.17. Conga LVM Graphical User Interface

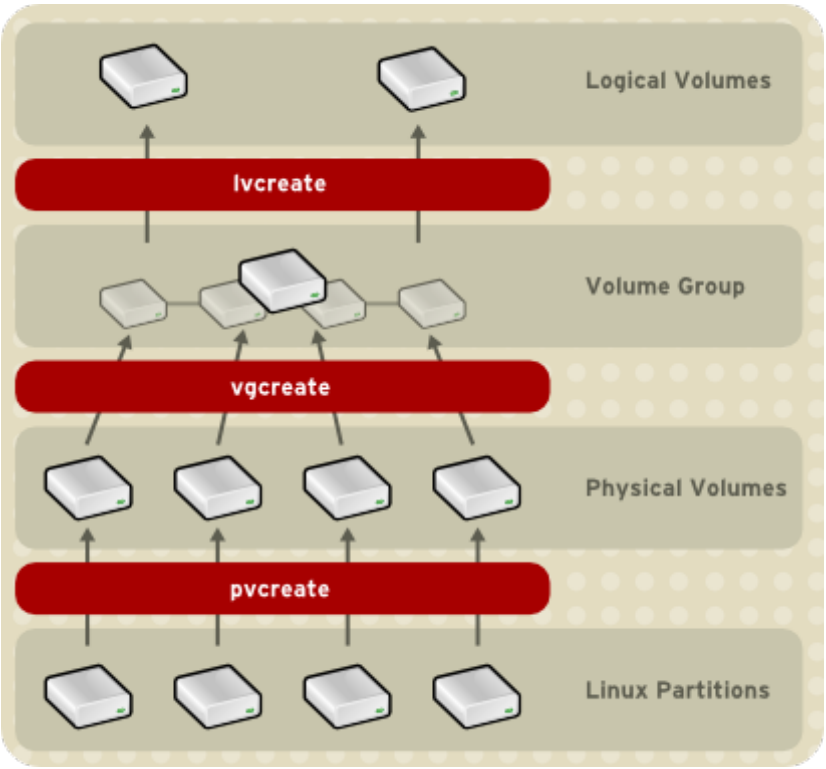

Figure 1.18. Creating Logical Volumes

1.7. Global Network Block Device

Global Network Block Device (GNBD) provides block-device access to Red Hat GFS over TCP/IP. GNBD is similar in concept to NBD; however, GNBD is GFS-specific and tuned solely for use with GFS. GNBD is useful when the need for more robust technologies — Fibre Channel or single-initiator SCSI — are not necessary or are cost-prohibitive.

GNBD consists of two major components: a GNBD client and a GNBD server. A GNBD client runs in a node with GFS and imports a block device exported by a GNBD server. A GNBD server runs in another node and exports block-level storage from its local storage (either directly attached storage or SAN storage). Refer to Figure 1.19, “GNBD Overview”. Multiple GNBD clients can access a device exported by a GNBD server, thus making a GNBD suitable for use by a group of nodes running GFS.

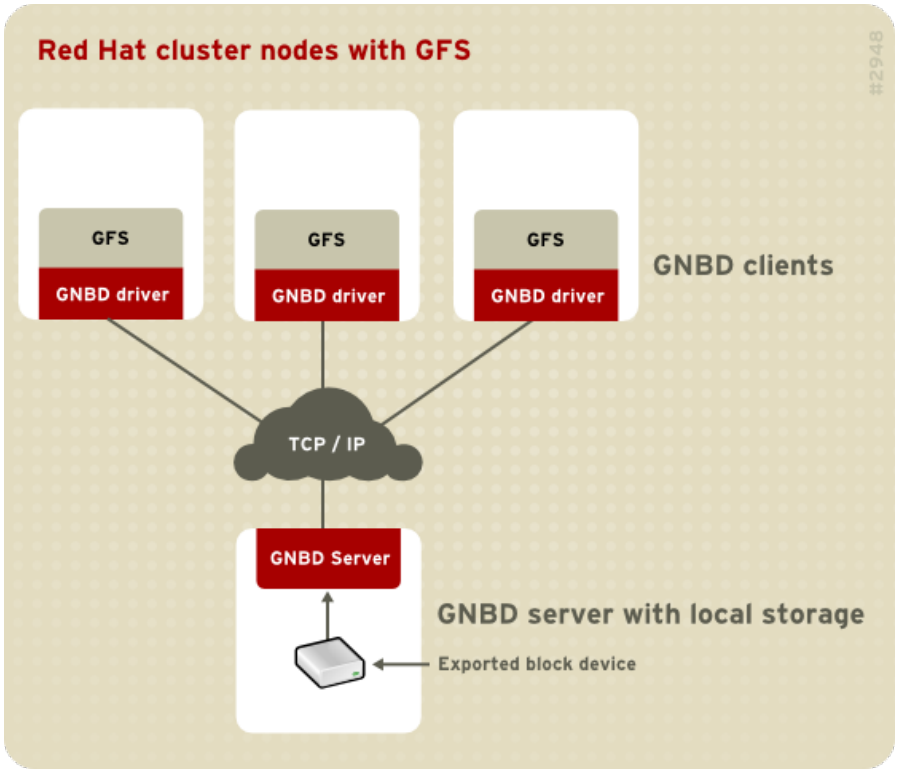

Figure 1.19. GNBD Overview

1.8. Linux Virtual Server

Linux Virtual Server (LVS) is a set of integrated software components for balancing the IP load across a set of real servers. LVS runs on a pair of equally configured computers: one that is an active LVS router and one that is a backup LVS router. The active LVS router serves two roles:

- To balance the load across the real servers.
- To check the integrity of the services on each real server.

The backup LVS router monitors the active LVS router and takes over from it in case the active LVS router fails.

Figure 1.20, “Components of a Running LVS Cluster” provides an overview of the LVS components and their interrelationship.

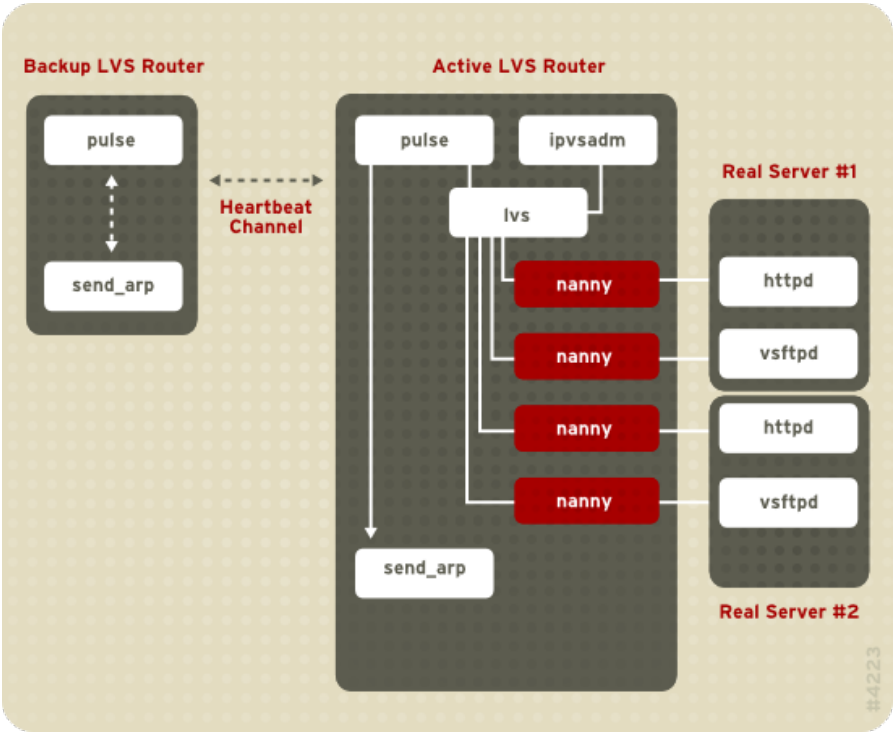

Figure 1.20. Components of a Running LVS Cluster

The **pulse** daemon runs on both the active and passive LVS routers. On the backup LVS router, **pulse** sends a *heartbeat* to the public interface of the active router to make sure the active LVS router is properly functioning. On the active LVS router, **pulse** starts the **lvs** daemon and responds to *heartbeat* queries from the backup LVS router.

Once started, the **lvs** daemon calls the **ipvsadm** utility to configure and maintain the IPVS (IP Virtual Server) routing table in the kernel and starts a **nanny** process for each configured virtual server on each real server. Each **nanny** process checks the state of one configured service on one real server, and tells the **lvs** daemon if the service on that real server is malfunctioning. If a malfunction is detected, the **lvs** daemon instructs **ipvsadm** to remove that real server from the IPVS routing table.

If the backup LVS router does not receive a response from the active LVS router, it initiates failover by calling **send_arp** to reassign all virtual IP addresses to the NIC hardware addresses (*MAC* address) of the backup LVS router, sends a command to the active LVS router via both the public and private network interfaces to shut down the **lvs** daemon on the active LVS router, and starts the **lvs** daemon on the backup LVS router to accept requests for the configured virtual servers.

To an outside user accessing a hosted service (such as a website or database application), LVS appears as one server. However, the user is actually accessing real servers behind the LVS routers.

Because there is no built-in component in LVS to share the data among real servers, you have have two basic options:

- Synchronize the data across the real servers.
- Add a third layer to the topology for shared data access.

The first option is preferred for servers that do not allow large numbers of users to upload or change data on the real servers. If the real servers allow large numbers of users to modify data, such as an e-commerce website, adding a third layer is preferable.

There are many ways to synchronize data among real servers. For example, you can use shell scripts to post updated web pages to the real servers simultaneously. Also, you can use programs such as **rsync** to replicate changed data across all nodes at a set interval. However, in environments where users frequently upload files or issue database transactions, using scripts or the **rsync** command for data synchronization does not function optimally. Therefore, for real servers with a high amount of uploads, database transactions, or similar traffic, a *three-tiered topology* is more appropriate for data synchronization.

1.8.1. Two-Tier LVS Topology

Figure 1.21, “Two-Tier LVS Topology” shows a simple LVS configuration consisting of two tiers: LVS routers and real servers. The LVS-router tier consists of one active LVS router and one backup LVS router. The real-server tier consists of real servers connected to the private network. Each LVS router has two network interfaces: one connected to a public network (Internet) and one connected to a private network. A network interface connected to each network allows the LVS routers to regulate traffic between clients on the public network and the real servers on the private network. In Figure 1.21, “Two-Tier LVS Topology”, the active LVS router uses *Network Address Translation* (*NAT*) to direct traffic from the public network to real servers on the private network, which in turn provide services as requested. The real servers pass all public traffic through the active LVS router. From the perspective of clients on the public network, the LVS router appears as one entity.

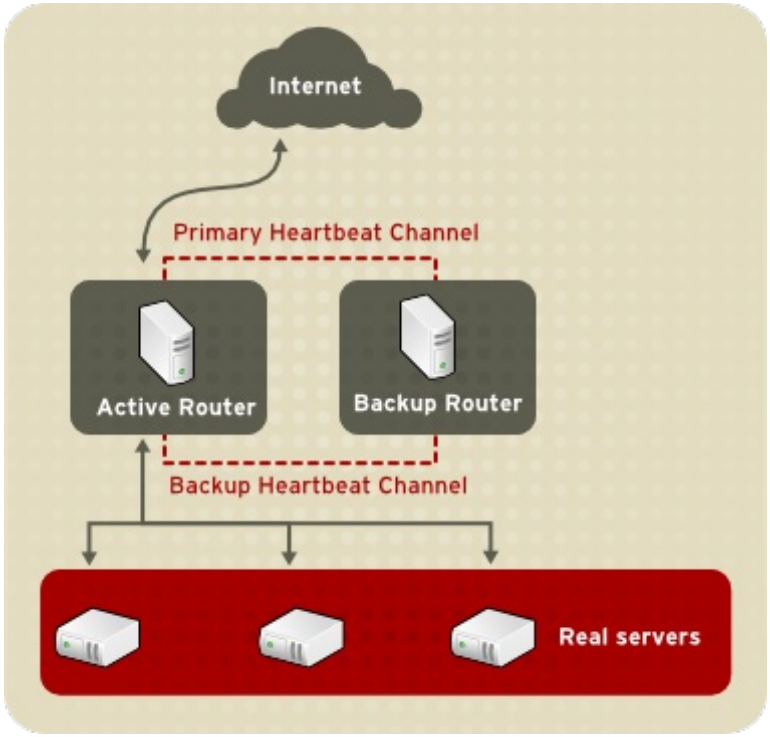

Figure 1.21. Two-Tier LVS Topology

Service requests arriving at an LVS router are addressed to a *virtual IP* address or VIP. This is a publicly-routable address that the administrator of the site associates with a fully-qualified domain name, such as www.example.com, and which is assigned to one or more *virtual servers* [1]. Note that a VIP address migrates from one LVS router to the other during a failover, thus maintaining a presence at that IP address, also known as *floating IP addresses*.

VIP addresses may be aliased to the same device that connects the LVS router to the public network. For instance, if eth0 is connected to the Internet, then multiple virtual servers can be aliased to **eth0:1**. Alternatively, each virtual server can be associated with a separate device per service. For example,

HTTP traffic can be handled on **eth0:1**, and FTP traffic can be handled on **eth0:2**.

Only one LVS router is active at a time. The role of the active LVS router is to redirect service requests from virtual IP addresses to the real servers. The redirection is based on one of eight load-balancing algorithms:

- Round-Robin Scheduling — Distributes each request sequentially around a pool of real servers. Using this algorithm, all the real servers are treated as equals without regard to capacity or load.
- Weighted Round-Robin Scheduling — Distributes each request sequentially around a pool of real servers but gives more jobs to servers with greater capacity. Capacity is indicated by a user-assigned weight factor, which is then adjusted up or down by dynamic load information. This is a preferred choice if there are significant differences in the capacity of real servers in a server pool. However, if the request load varies dramatically, a more heavily weighted server may answer more than its share of requests.
- Least-Connection — Distributes more requests to real servers with fewer active connections. This is a type of dynamic scheduling algorithm, making it a better choice if there is a high degree of variation in the request load. It is best suited for a real server pool where each server node has roughly the same capacity. If the real servers have varying capabilities, weighted least-connection scheduling is a better choice.
- Weighted Least-Connections (default) — Distributes more requests to servers with fewer active connections relative to their capacities. Capacity is indicated by a user-assigned weight, which is then adjusted up or down by dynamic load information. The addition of weighting makes this algorithm ideal when the real server pool contains hardware of varying capacity.
- Locality-Based Least-Connection Scheduling — Distributes more requests to servers with fewer active connections relative to their destination IPs. This algorithm is for use in a proxy-cache server cluster. It routes the packets for an IP address to the server for that address unless that server is above its capacity and has a server in its half load, in which case it assigns the IP address to the least loaded real server.
- Locality-Based Least-Connection Scheduling with Replication Scheduling — Distributes more requests to servers with fewer active connections relative to their destination IPs. This algorithm is also for use in a proxy-cache server cluster. It differs from Locality-Based Least-Connection Scheduling by mapping the target IP address to a subset of real server nodes. Requests are then routed to the server in this subset with the lowest number of connections. If all the nodes for the destination IP are above capacity, it replicates a new server for that destination IP address by adding the real server with the least connections from the overall pool of real servers to the subset of real servers for that destination IP. The most-loaded node is then dropped from the real server subset to prevent over-replication.
- Source Hash Scheduling — Distributes requests to the pool of real servers by looking up the source IP in a static hash table. This algorithm is for LVS routers with multiple firewalls.

Also, the active LVS router dynamically monitors the overall health of the specific services on the real servers through simple *send/expect scripts*. To aid in detecting the health of services that require dynamic data, such as HTTPS or SSL, you can also call external executables. If a service on a real server malfunctions, the active LVS router stops sending jobs to that server until it returns to normal operation.

The backup LVS router performs the role of a standby system. Periodically, the LVS routers exchange heartbeat messages through the primary external public interface and, in a failover situation, the private interface. Should the backup LVS router fail to receive a heartbeat message within an expected interval, it initiates a failover and assumes the role of the active LVS router. During failover, the backup LVS router takes over the VIP addresses serviced by the failed router using a technique known as *ARP spoofing* — where the backup LVS router announces itself as the destination for IP packets addressed to the failed node. When the failed node returns to active service, the backup LVS router assumes its backup role

again.

The simple, two-tier configuration in Figure 1.21, “Two-Tier LVS Topology” is suited best for clusters serving data that does not change very frequently — such as static web pages — because the individual real servers do not automatically synchronize data among themselves.

1.8.2. Three-Tier LVS Topology

Figure 1.22, “Three-Tier LVS Topology” shows a typical three-tier LVS configuration. In the example, the active LVS router routes the requests from the public network (Internet) to the second tier — real servers. Each real server then accesses a shared data source of a Red Hat cluster in the third tier over the private network.

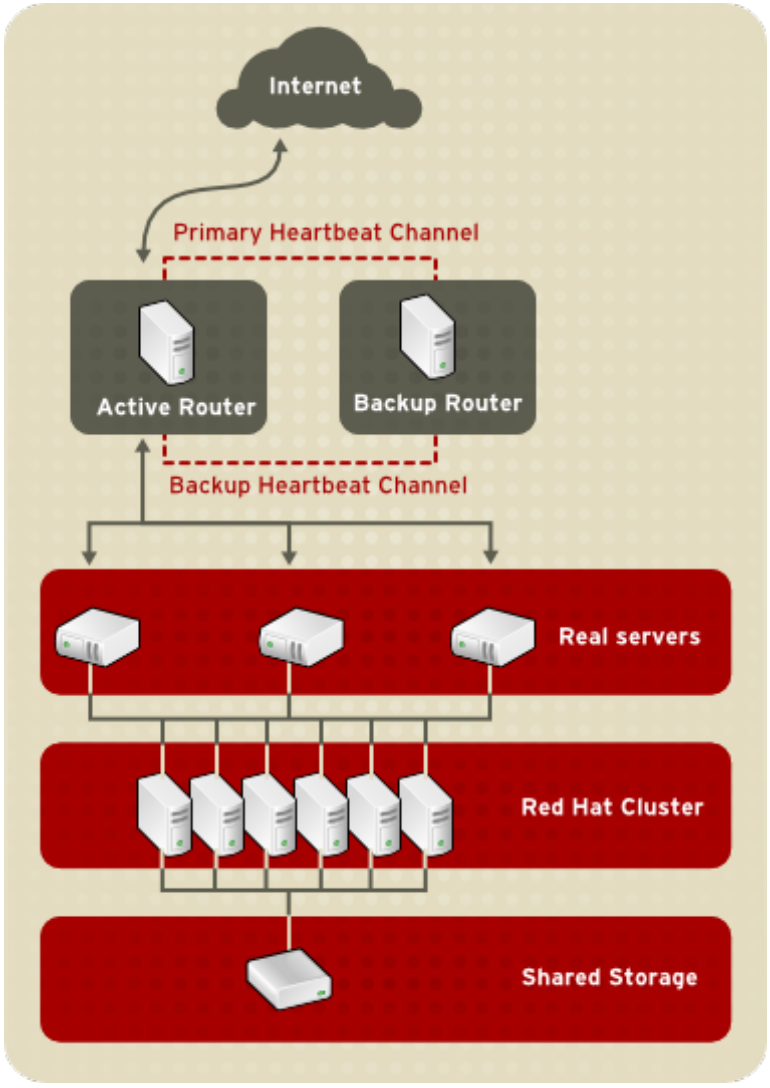

Figure 1.22. Three-Tier LVS Topology

This topology is suited well for busy FTP servers, where accessible data is stored on a central, highly available server and accessed by each real server via an exported NFS directory or Samba share. This topology is also recommended for websites that access a central, high-availability database for transactions. Additionally, using an active-active configuration with a Red Hat cluster, you can configure one high-availability cluster to serve both of these roles simultaneously.

1.8.3. Routing Methods

You can use Network Address Translation (NAT) routing or direct routing with LVS. The following sections briefly describe NAT routing and direct routing with LVS.

1.8.3.1. NAT Routing

Figure 1.23, “LVS Implemented with NAT Routing”, illustrates LVS using NAT routing to move requests between the Internet and a private network.

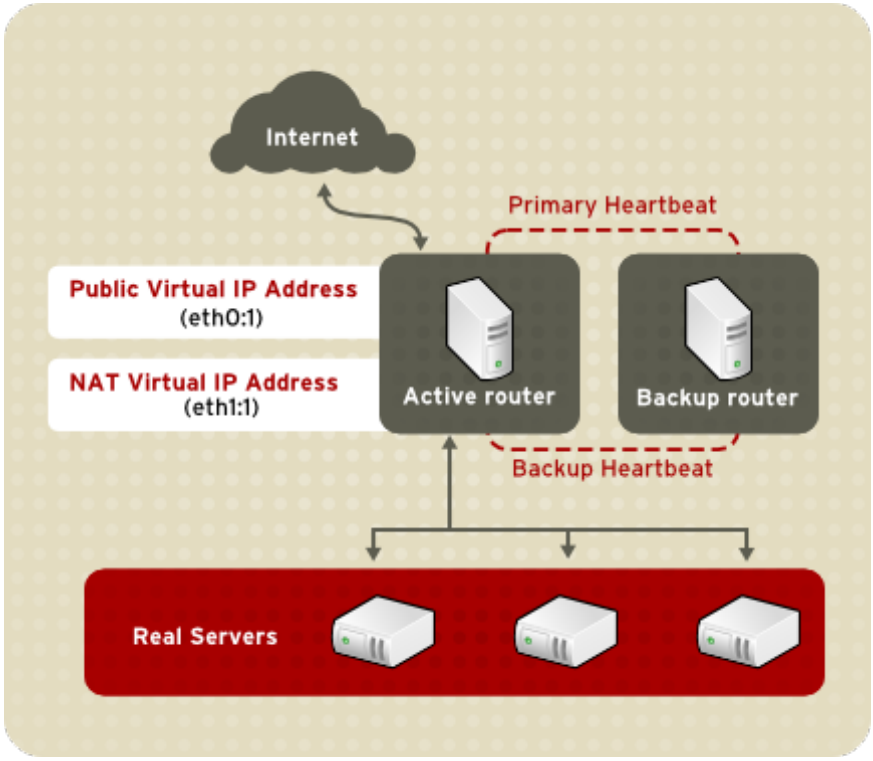

Figure 1.23. LVS Implemented with NAT Routing

In the example, there are two NICs in the active LVS router. The NIC for the Internet has a *real IP address* on eth0 and has a floating IP address aliased to eth0:1. The NIC for the private network interface has a real IP address on eth1 and has a floating IP address aliased to eth1:1. In the event of failover, the virtual interface facing the Internet and the private facing virtual interface are taken over by the backup LVS router simultaneously. All the real servers on the private network use the floating IP for the NAT router as their default route to communicate with the active LVS router so that their abilities to respond to requests from the Internet is not impaired.

In the example, the LVS router's public LVS floating IP address and private NAT floating IP address are aliased to two physical NICs. While it is possible to associate each floating IP address to its physical device on the LVS router nodes, having more than two NICs is not a requirement.

Using this topology, the active LVS router receives the request and routes it to the appropriate server. The real server then processes the request and returns the packets to the LVS router. The LVS router uses network address translation to replace the address of the real server in the packets with the LVS routers public VIP address. This process is called *IP masquerading* because the actual IP addresses of the real servers is hidden from the requesting clients.

Using NAT routing, the real servers can be any kind of computers running a variety operating systems. The main disadvantage of NAT routing is that the LVS router may become a bottleneck in large deployments because it must process outgoing and incoming requests.

1.8.3.2. Direct Routing

Direct routing provides increased performance benefits compared to NAT routing. Direct routing allows the real servers to process and route packets directly to a requesting user rather than passing outgoing packets through the LVS router. Direct routing reduces the possibility of network performance issues by relegating the job of the LVS router to processing incoming packets only.

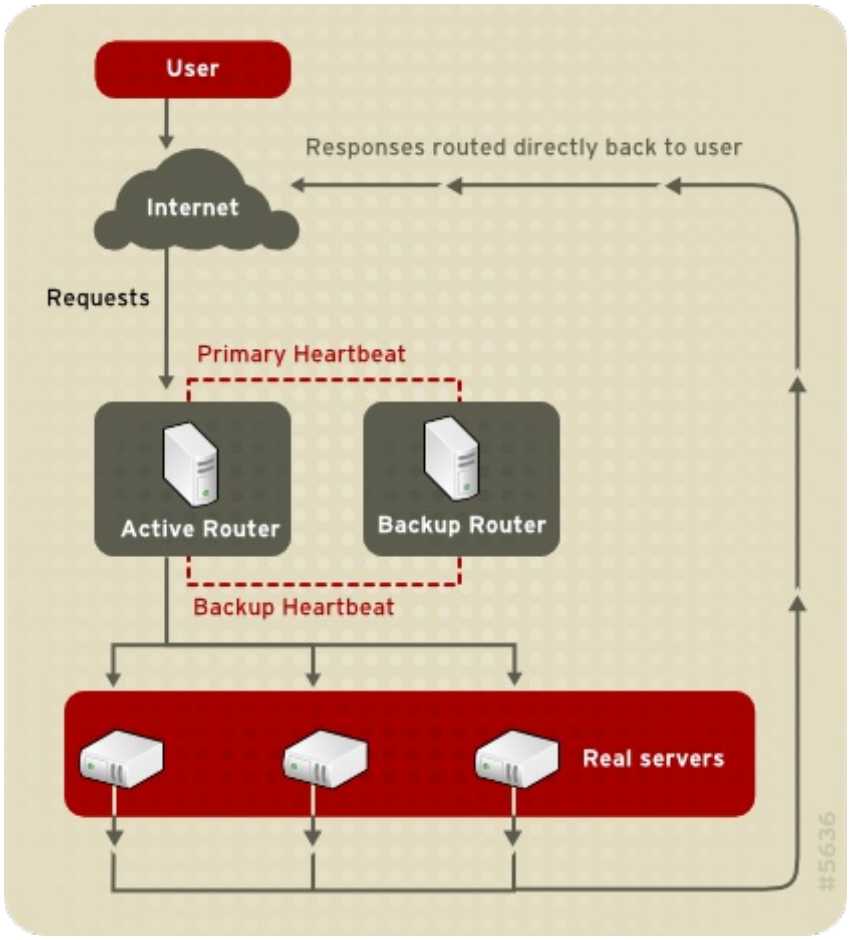

Figure 1.24. LVS Implemented with Direct Routing

In a typical direct-routing LVS configuration, an LVS router receives incoming server requests through a virtual IP (VIP) and uses a scheduling algorithm to route the request to real servers. Each real server processes requests and sends responses directly to clients, bypassing the LVS routers. Direct routing allows for scalability in that real servers can be added without the added burden on the LVS router to route outgoing packets from the real server to the client, which can become a bottleneck under heavy network load.

While there are many advantages to using direct routing in LVS, there are limitations. The most common issue with direct routing and LVS is with *Address Resolution Protocol* (ARP).

In typical situations, a client on the Internet sends a request to an IP address. Network routers typically send requests to their destination by relating IP addresses to a machine's MAC address with ARP. ARP requests are broadcast to all connected machines on a network, and the machine with the correct IP/MAC address combination receives the packet. The IP/MAC associations are stored in an ARP cache, which is cleared periodically (usually every 15 minutes) and refilled with IP/MAC associations.

The issue with ARP requests in a direct-routing LVS configuration is that because a client request to an IP address must be associated with a MAC address for the request to be handled, the virtual IP address of the LVS router must also be associated to a MAC. However, because both the LVS router and the real

servers have the same VIP, the ARP request is broadcast to all the nodes associated with the VIP. This can cause several problems, such as the VIP being associated directly to one of the real servers and processing requests directly, bypassing the LVS router completely and defeating the purpose of the LVS configuration. Using an LVS router with a powerful CPU that can respond quickly to client requests does not necessarily remedy this issue. If the LVS router is under heavy load, it may respond to the ARP request more slowly than an underutilized real server, which responds more quickly and is assigned the VIP in the ARP cache of the requesting client.

To solve this issue, the incoming requests should *only* associate the VIP to the LVS router, which will properly process the requests and send them to the real server pool. This can be done by using the **`arptables`** packet-filtering tool.

1.8.4. Persistence and Firewall Marks

In certain situations, it may be desirable for a client to reconnect repeatedly to the same real server, rather than have an LVS load-balancing algorithm send that request to the best available server. Examples of such situations include multi-screen web forms, cookies, SSL, and FTP connections. In those cases, a client may not work properly unless the transactions are being handled by the same server to retain context. LVS provides two different features to handle this: *persistence* and *firewall marks*.

1.8.4.1. Persistence

When enabled, persistence acts like a timer. When a client connects to a service, LVS remembers the last connection for a specified period of time. If that same client IP address connects again within that period, it is sent to the same server it connected to previously — bypassing the load-balancing mechanisms. When a connection occurs outside the time window, it is handled according to the scheduling rules in place.

Persistence also allows you to specify a subnet mask to apply to the client IP address test as a tool for controlling what addresses have a higher level of persistence, thereby grouping connections to that subnet.

Grouping connections destined for different ports can be important for protocols that use more than one port to communicate, such as FTP. However, persistence is not the most efficient way to deal with the problem of grouping together connections destined for different ports. For these situations, it is best to use *firewall marks*.

1.8.4.2. Firewall Marks

Firewall marks are an easy and efficient way to a group ports used for a protocol or group of related protocols. For example, if LVS is deployed to run an e-commerce site, firewall marks can be used to bundle HTTP connections on port 80 and secure, HTTPS connections on port 443. By assigning the same firewall mark to the virtual server for each protocol, state information for the transaction can be preserved because the LVS router forwards all requests to the same real server after a connection is opened.

Because of its efficiency and ease-of-use, administrators of LVS should use firewall marks instead of persistence whenever possible for grouping connections. However, you should still add persistence to the virtual servers in conjunction with firewall marks to ensure the clients are reconnected to the same server for an adequate period of time.

1.9. Cluster Administration Tools

Red Hat Cluster Suite provides a variety of tools to configure and manage your Red Hat Cluster. This section provides an overview of the administration tools available with Red Hat Cluster Suite:

- Section 1.9.1, “Conga”
- Section 1.9.2, “Cluster Administration GUI”
- Section 1.9.3, “Command Line Administration Tools”

1.9.1. Conga

Conga is an integrated set of software components that provides centralized configuration and management of Red Hat clusters and storage. **Conga** provides the following major features:

- One Web interface for managing cluster and storage
- Automated Deployment of Cluster Data and Supporting Packages
- Easy Integration with Existing Clusters
- No Need to Re-Authenticate
- Integration of Cluster Status and Logs
- Fine-Grained Control over User Permissions

The primary components in **Conga** are **luci** and **ricci**, which are separately installable. **luci** is a server that runs on one computer and communicates with multiple clusters and computers via **ricci**. **ricci** is an agent that runs on each computer (either a cluster member or a standalone computer) managed by **Conga**.

luci is accessible through a Web browser and provides three major functions that are accessible through the following tabs:

- **homebase** — Provides tools for adding and deleting computers, adding and deleting users, and configuring user privileges. Only a system administrator is allowed to access this tab.
- **cluster** — Provides tools for creating and configuring clusters. Each instance of **luci** lists clusters that have been set up with that **luci**. A system administrator can administer all clusters listed on this tab. Other users can administer only clusters that the user has permission to manage (granted by an administrator).
- **storage** — Provides tools for remote administration of storage. With the tools on this tab, you can manage storage on computers whether they belong to a cluster or not.

To administer a cluster or storage, an administrator adds (or *registers*) a cluster or a computer to a **luci** server. When a cluster or a computer is registered with **luci**, the FQDN hostname or IP address of each computer is stored in a **luci** database.

You can populate the database of one **luci** instance from another **luci**instance. That capability provides a means of replicating a **luci** server instance and provides an efficient upgrade and testing path. When you install an instance of **luci**, its database is empty. However, you can import part or all of a **luci** database from an existing **luci** server when deploying a new **luci** server.

Each **luci** instance has one user at initial installation — admin. Only the admin user may add systems to a **luci** server. Also, the admin user can create additional user accounts and determine which users are allowed to access clusters and computers registered in the **luci** database. It is possible to import users as a batch operation in a new **luci** server, just as it is possible to import clusters and computers.

When a computer is added to a **luci** server to be administered, authentication is done once. No authentication is necessary from then on (unless the certificate used is revoked by a CA). After that, you can remotely configure and manage clusters and storage through the **luci** user interface. **luci** and **ricci** communicate with each other via XML.

The following figures show sample displays of the three major **luci** tabs: **homebase**, **cluster**, and

storage.

For more information about **Conga**, refer to *Configuring and Managing a Red Hat Cluster* and the online help available with the **luci** server.

Figure 1.25. luci homebase Tab

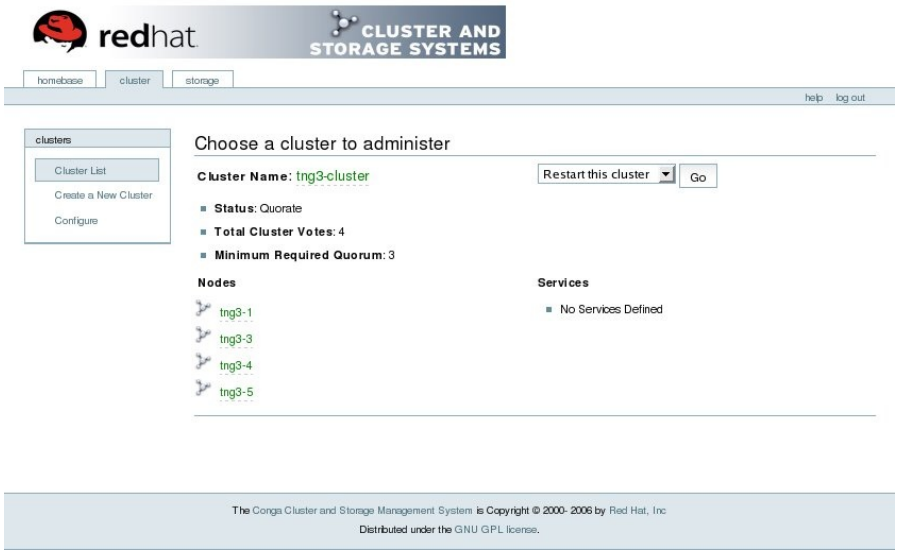

Figure 1.26. luci cluster Tab

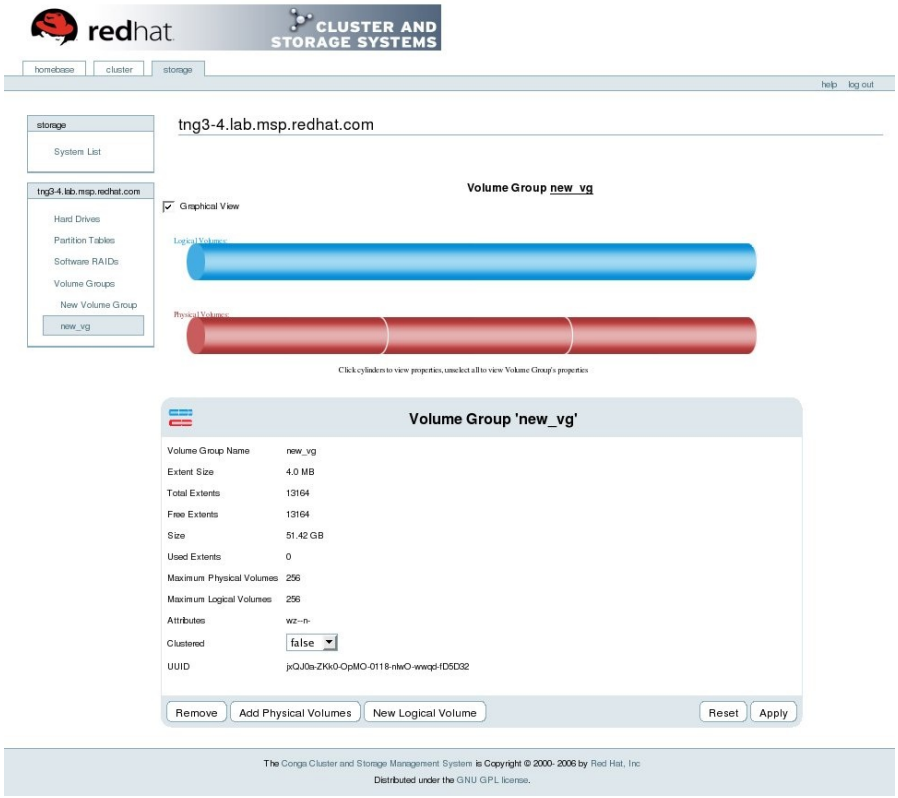

Figure 1.27. luci storage Tab

1.9.2. Cluster Administration GUI

This section provides an overview of the **system-config-cluster** cluster administration graphical user interface (GUI) available with Red Hat Cluster Suite. The GUI is for use with the cluster infrastructure and the high-availability service management components (refer to Section 1.3, “Cluster Infrastructure” and Section 1.4, “High-availability Service Management”). The GUI consists of two major functions: the **Cluster Configuration Tool** and the **Cluster Status Tool**. The **Cluster Configuration Tool** provides the capability to create, edit, and propagate the cluster configuration file (**/etc/cluster/cluster.conf**). The **Cluster Status Tool** provides the capability to manage high-availability services. The following sections summarize those functions.

- Section 1.9.2.1, “**Cluster Configuration Tool**”
- Section 1.9.2.2, “**Cluster Status Tool**”

1.9.2.1. Cluster Configuration Tool

You can access the **Cluster Configuration Tool** (Figure 1.28, “Cluster Configuration Tool”) through the **Cluster Configuration** tab in the Cluster Administration GUI.

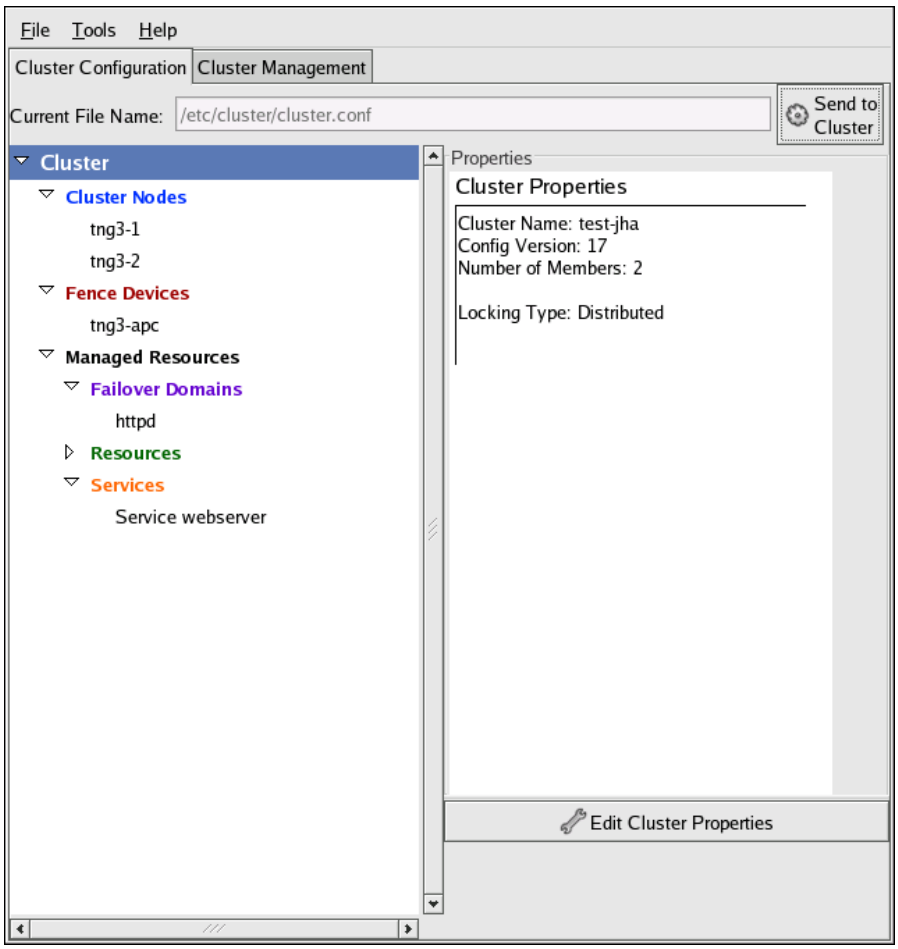

Figure 1.28. Cluster Configuration Tool

The **Cluster Configuration Tool** represents cluster configuration components in the configuration file (**/etc/cluster/cluster.conf**) with a hierarchical graphical display in the left panel. A triangle icon to the left of a component name indicates that the component has one or more subordinate components assigned to it. Clicking the triangle icon expands and collapses the portion of the tree below a component. The components displayed in the GUI are summarized as follows:

- **Cluster Nodes** — Displays cluster nodes. Nodes are represented by name as subordinate elements under **Cluster Nodes**. Using configuration buttons at the bottom of the right frame (below **Properties**), you can add nodes, delete nodes, edit node properties, and configure fencing methods for each node.
- **Fence Devices** — Displays fence devices. Fence devices are represented as subordinate elements under **Fence Devices**. Using configuration buttons at the bottom of the right frame (below **Properties**), you can add fence devices, delete fence devices, and edit fence-device properties. Fence devices must be defined before you can configure fencing (with the **Manage Fencing For This Node** button) for each node.
- **Managed Resources** — Displays failover domains, resources, and services.
  - **Failover Domains** — For configuring one or more subsets of cluster nodes used to run a high-availability service in the event of a node failure. Failover domains are represented as subordinate elements under **Failover Domains**. Using configuration buttons at the bottom of the right frame (below **Properties**), you can create failover domains (when **Failover Domains** is selected) or edit failover domain properties (when a failover domain is selected).
  - **Resources** — For configuring shared resources to be used by high-availability services. Shared

resources consist of file systems, IP addresses, NFS mounts and exports, and user-created scripts that are available to any high-availability service in the cluster. Resources are represented as subordinate elements under **Resources**. Using configuration buttons at the bottom of the right frame (below **Properties**), you can create resources (when **Resources** is selected) or edit resource properties (when a resource is selected).

Note

The **Cluster Configuration Tool** provides the capability to configure private resources, also. A private resource is a resource that is configured for use with only one service. You can configure a private resource within a **Service** component in the GUI.

- **Services** — For creating and configuring high-availability services. A service is configured by assigning resources (shared or private), assigning a failover domain, and defining a recovery policy for the service. Services are represented as subordinate elements under **Services**. Using configuration buttons at the bottom of the right frame (below **Properties**), you can create services (when **Services** is selected) or edit service properties (when a service is selected).

1.9.2.2. Cluster Status Tool

You can access the **Cluster Status Tool** (Figure 1.29, “Cluster Status Tool”) through the **Cluster Management** tab in Cluster Administration GUI.

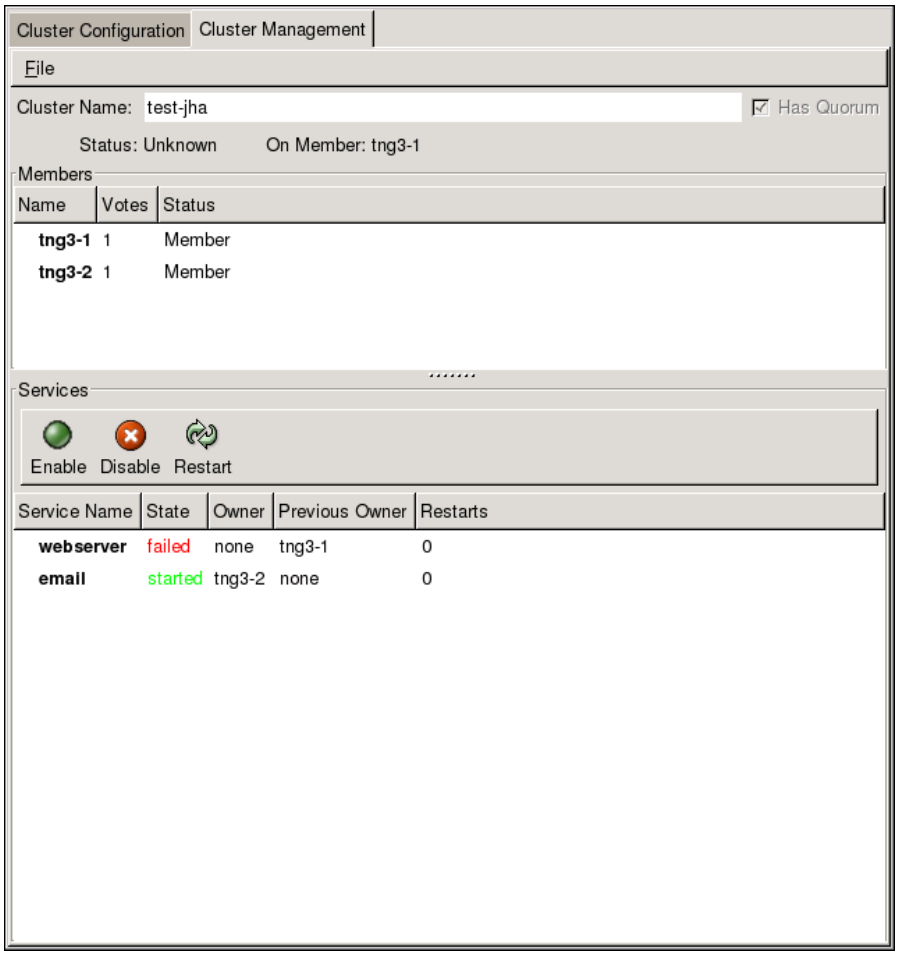

Figure 1.29. Cluster Status Tool

The nodes and services displayed in the **Cluster Status Tool** are determined by the cluster configuration file (**/etc/cluster/cluster.conf**). You can use the **Cluster Status Tool** to enable, disable, restart, or relocate a high-availability service.

1.9.3. Command Line Administration Tools

In addition to **Conga** and the **system-config-cluster** Cluster Administration GUI, command line tools are available for administering the cluster infrastructure and the high-availability service management components. The command line tools are used by the Cluster Administration GUI and init scripts supplied by Red Hat. Table 1.1, "Command Line Tools" summarizes the command line tools.

Table 1.1. Command Line Tools

Command Line Tool	Used With	Purpose
ccs_tool — Cluster Configuration System Tool	Cluster Infrastructure	**ccs_tool** is a program for making online updates to the cluster configuration file. It provides the capability to create and modify cluster infrastructure components (for example, creating a cluster, adding and removing a node). For more information about this tool, refer to the ccs_tool(8) man page.
cman_tool — Cluster Management Tool	Cluster Infrastructure	**cman_tool** is a program that manages the CMAN cluster manager. It provides the capability to join a cluster, leave a cluster, kill a node, or change the expected quorum votes of a node in a cluster. **cman_tool** is available with DLM clusters only. For more information about this tool, refer to the cman_tool(8) man page.
gulm_tool — Cluster Management Tool	Cluster Infrastructure	**gulm_tool** is a program used to manage GULM. It provides an interface to **lock_gulmd**, the GULM lock manager. **gulm_tool** is available with GULM clusters only. For more information about this tool, refer to the gulm_tool(8) man page.
fence_tool — Fence Tool	Cluster Infrastructure	**fence_tool** is a program used to join or leave the default fence domain. Specifically, it starts the fence daemon (**fenced**) to join the domain and kills **fenced** to leave the domain. **fence_tool** is available with DLM clusters only. For more information about this tool, refer to the fence_tool(8) man page.
clustat — Cluster Status Utility	High-availability Service Management Components	The **clustat** command displays the status of the cluster. It shows membership information, quorum view, and the state of all configured user services. For more information about this tool, refer to the clustat(8) man page.
clusvcadm — Cluster User Service Administration Utility	High-availability Service Management Components	The **clusvcadm** command allows you to enable, disable, relocate, and restart high-availability services in a cluster. For more information about this tool, refer to the clusvcadm(8) man page.

1.10. Linux Virtual Server Administration GUI

This section provides an overview of the LVS configuration tool available with Red Hat Cluster Suite — the **Piranha Configuration Tool**. The **Piranha Configuration Tool** is a Web-browser graphical user interface (GUI) that provides a structured approach to creating the configuration file for LVS — **/etc/sysconfig/ha/lvs.cf**.

To access the **Piranha Configuration Tool** you need the **piranha-gui** service running on the active LVS router. You can access the **Piranha Configuration Tool** locally or remotely with a Web browser. You can access it locally with this URL: **http://localhost:3636**. You can access it remotely with either the hostname or the real IP address followed by **:3636**. If you are accessing the **Piranha Configuration Tool** remotely, you need an **ssh** connection to the active LVS router as the root user.

Starting the **Piranha Configuration Tool** causes the **Piranha Configuration Tool** welcome page to be displayed (refer to Figure 1.30, “The Welcome Panel”). Logging in to the welcome page provides access to the four main screens or *panels*: **CONTROL/MONITORING**, **GLOBAL SETTINGS**, **REDUNDANCY**, and **VIRTUAL SERVERS**. In addition, the **VIRTUAL SERVERS** panel contains four *subsections*. The **CONTROL/MONITORING** panel is the first panel displayed after you log in at the welcome screen.

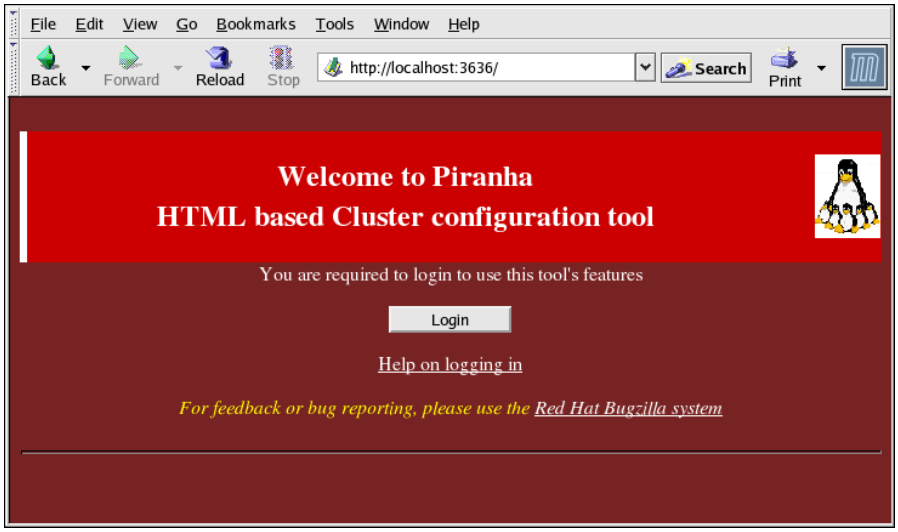
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/
Search
Print
Welcome to Piranha
HTML based Cluster configuration tool
You are required to login to use this tool's features
Login
Help on logging in
For feedback or bug reporting, please use the Red Hat Bugzilla system

Figure 1.30. The Welcome Panel

The following sections provide a brief description of the **Piranha Configuration Tool** configuration pages.

1.10.1. CONTROL/MONITORING

The **CONTROL/MONITORING** Panel displays runtime status. It displays the status of the **pulse** daemon, the LVS routing table, and the LVS-spawned **nanny** processes.

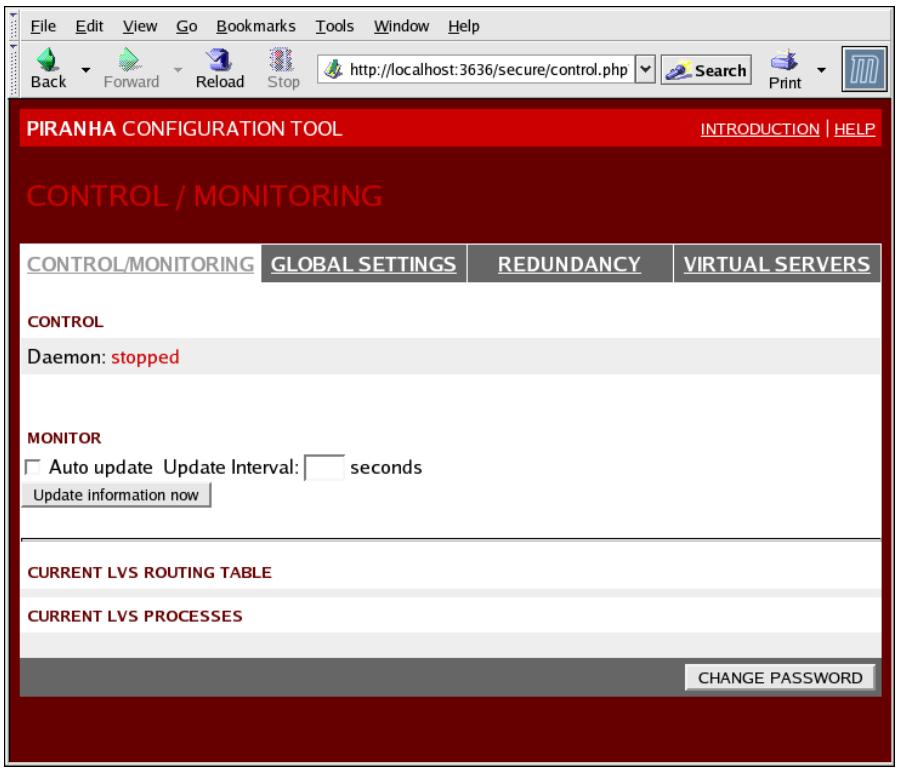

Figure 1.31. The CONTROL/MONITORING Panel

Auto update

Enables the status display to be updated automatically at a user-configurable interval set in the **Update frequency in seconds** text box (the default value is 10 seconds).

It is not recommended that you set the automatic update to an interval less than 10 seconds. Doing so may make it difficult to reconfigure the **Auto update** interval because the page will update too frequently. If you encounter this issue, simply click on another panel and then back on **CONTROL/MONITORING**.

Update information now

Provides manual update of the status information.

CHANGE PASSWORD

Clicking this button takes you to a help screen with information on how to change the administrative password for the **Piranha Configuration Tool**.

1.10.2. GLOBAL SETTINGS

The **GLOBAL SETTINGS** panel is where the LVS administrator defines the networking details for the primary LVS router's public and private network interfaces.

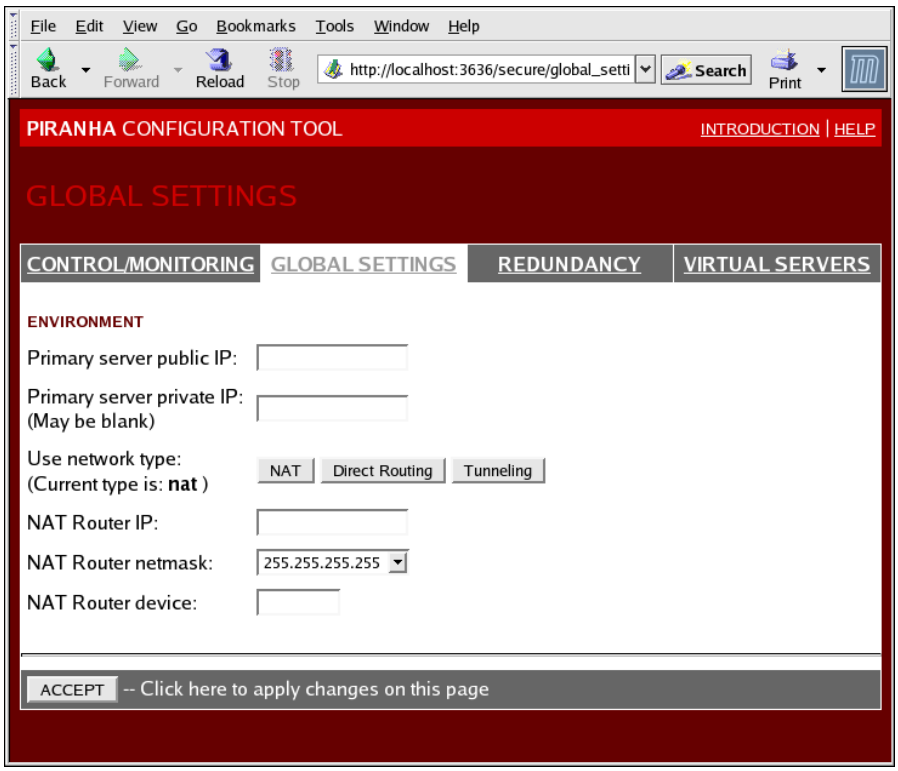

Figure 1.32. The GLOBAL SETTINGS Panel

The top half of this panel sets up the primary LVS router's public and private network interfaces.

Primary server public IP

The publicly routable real IP address for the primary LVS node.

Primary server private IP

The real IP address for an alternative network interface on the primary LVS node. This address is used solely as an alternative heartbeat channel for the backup router.

Use network type

Selects select NAT routing.

The next three fields are specifically for the NAT router's virtual network interface connected the private network with the real servers.

NAT Router IP

The private floating IP in this text field. This floating IP should be used as the gateway for the real servers.

NAT Router netmask

If the NAT router's floating IP needs a particular netmask, select it from drop-down list.

NAT Router device

Defines the device name of the network interface for the floating IP address, such as **eth1:1**.

1.10.3. REDUNDANCY

The **REDUNDANCY** panel allows you to configure of the backup LVS router node and set various heartbeat monitoring options.

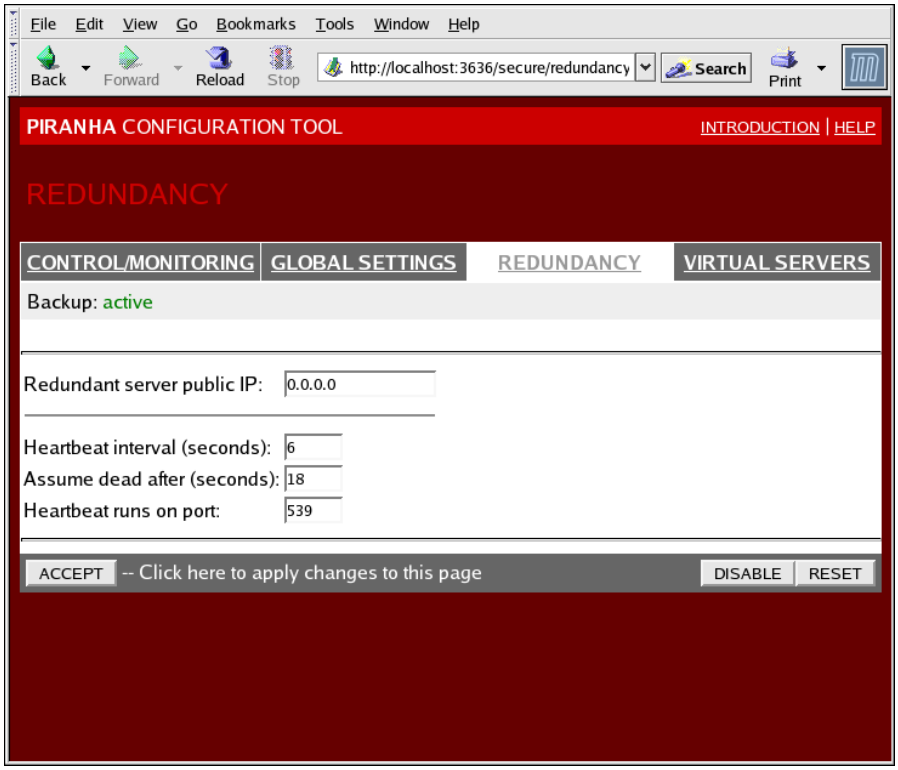

Figure 1.33. The REDUNDANCY Panel

Redundant server public IP

The public real IP address for the backup LVS router.

Redundant server private IP

The backup router's private real IP address.

The rest of the panel is for configuring the heartbeat channel, which is used by the backup node to monitor the primary node for failure.

Heartbeat Interval (seconds)

Sets the number of seconds between heartbeats — the interval that the backup node will check the functional status of the primary LVS node.

Assume dead after (seconds)

If the primary LVS node does not respond after this number of seconds, then the backup LVS router node will initiate failover.

Heartbeat runs on port

Sets the port at which the heartbeat communicates with the primary LVS node. The default is set to 539 if this field is left blank.

1.10.4. VIRTUAL SERVERS

The **VIRTUAL SERVERS** panel displays information for each currently defined virtual server. Each table entry shows the status of the virtual server, the server name, the virtual IP assigned to the server, the netmask of the virtual IP, the port number to which the service communicates, the protocol used, and the virtual device interface.

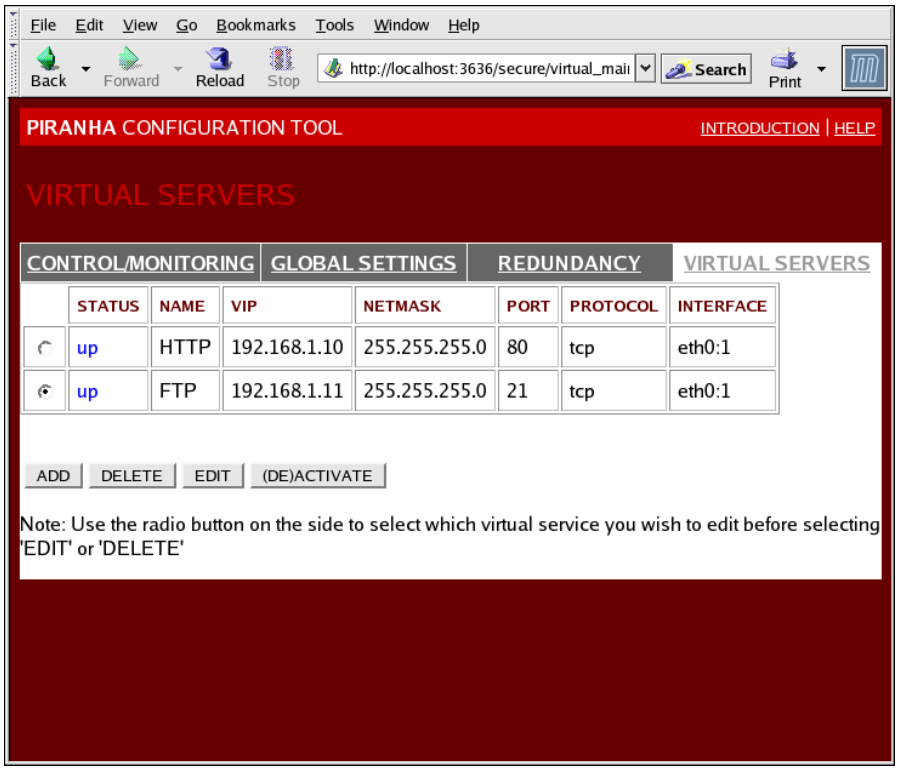
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_mai
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
VIRTUAL SERVERS
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
STATUS NAME VIP NETMASK PORT PROTOCOL INTERFACE
up HTTP 192.168.1.10 255.255.255.0 80 tcp eth0:1
up FTP 192.168.1.11 255.255.255.0 21 tcp eth0:1
ADD DELETE EDIT (DE)ACTIVATE
Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

Figure 1.34. The VIRTUAL SERVERS Panel

Each server displayed in the **VIRTUAL SERVERS** panel can be configured on subsequent screens or *subsections*.

To add a service, click the **ADD** button. To remove a service, select it by clicking the radio button next to the virtual server and click the **DELETE** button.

To enable or disable a virtual server in the table click its radio button and click the **(DE)ACTIVATE** button.

After adding a virtual server, you can configure it by clicking the radio button to its left and clicking the **EDIT** button to display the **VIRTUAL SERVER** subsection.

1.10.4.1. The VIRTUAL SERVER Subsection

The **VIRTUAL SERVER** subsection panel shown in Figure 1.35, “The **VIRTUAL SERVERS** Subsection” allows you to configure an individual virtual server. Links to subsections related specifically to this virtual server are located along the top of the page. But before configuring any of the subsections related to this virtual server, complete this page and click on the **ACCEPT** button.

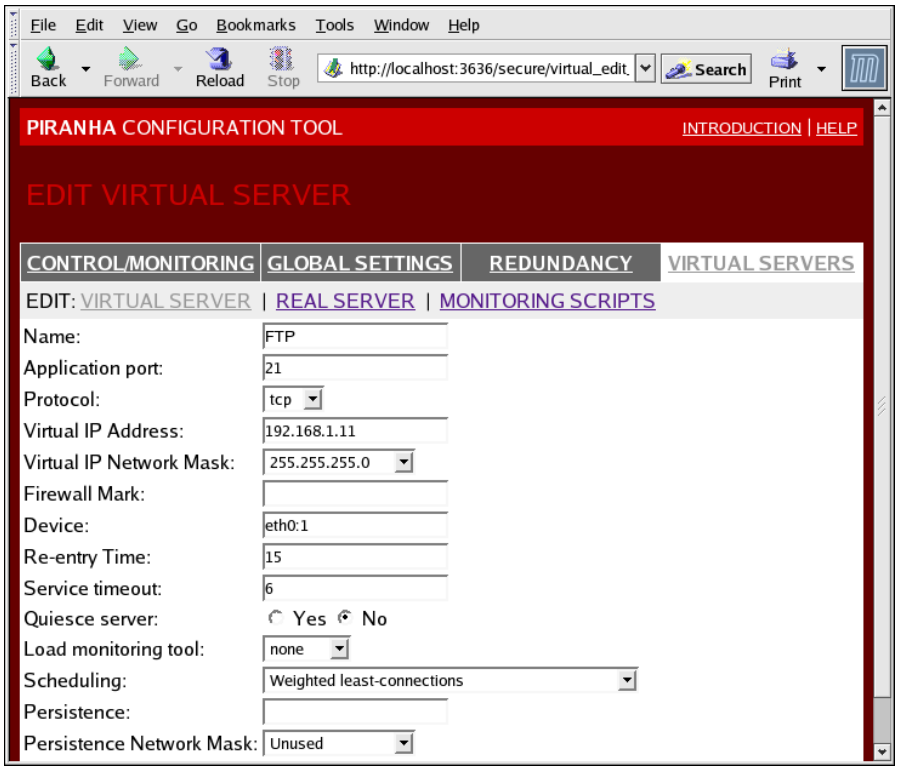

Figure 1.35. The VIRTUAL SERVERS Subsection

Name

A descriptive name to identify the virtual server. This name is *not* the hostname for the machine, so make it descriptive and easily identifiable. You can even reference the protocol used by the virtual server, such as HTTP.

Application port

The port number through which the service application will listen.

Protocol

Provides a choice of UDP or TCP, in a drop-down menu.

Virtual IP Address

The virtual server's floating IP address.

Virtual IP Network Mask

The netmask for this virtual server, in the drop-down menu.

Firewall Mark

For entering a firewall mark integer value when bundling multi-port protocols or creating a multi-port virtual server for separate, but related protocols.

Device

The name of the network device to which you want the floating IP address defined in the

Virtual IP Address field to bind.

You should alias the public floating IP address to the Ethernet interface connected to the public network.

Re-entry Time

An integer value that defines the number of seconds before the active LVS router attempts to use a real server after the real server failed.

Service Timeout

An integer value that defines the number of seconds before a real server is considered dead and not available.

Quiesce server

When the **Quiesce server** radio button is selected, anytime a new real server node comes online, the least-connections table is reset to zero so the active LVS router routes requests as if all the real servers were freshly added to the cluster. This option prevents the a new server from becoming bogged down with a high number of connections upon entering the cluster.

Load monitoring tool

The LVS router can monitor the load on the various real servers by using either **rup** or **ruptime**. If you select **rup** from the drop-down menu, each real server must run the **rstatd** service. If you select **ruptime**, each real server must run the **rwhod** service.

Scheduling

The preferred scheduling algorithm from the drop-down menu. The default is **Weighted least-connection**.

Persistence

Used if you need persistent connections to the virtual server during client transactions. Specifies the number of seconds of inactivity allowed to lapse before a connection times out in this text field.

Persistence Network Mask

To limit persistence to particular subnet, select the appropriate network mask from the drop-down menu.

1.10.4.2. REAL SERVER Subsection

Clicking on the **REAL SERVER** subsection link at the top of the panel displays the **EDIT REAL SERVER** subsection. It displays the status of the physical server hosts for a particular virtual service.

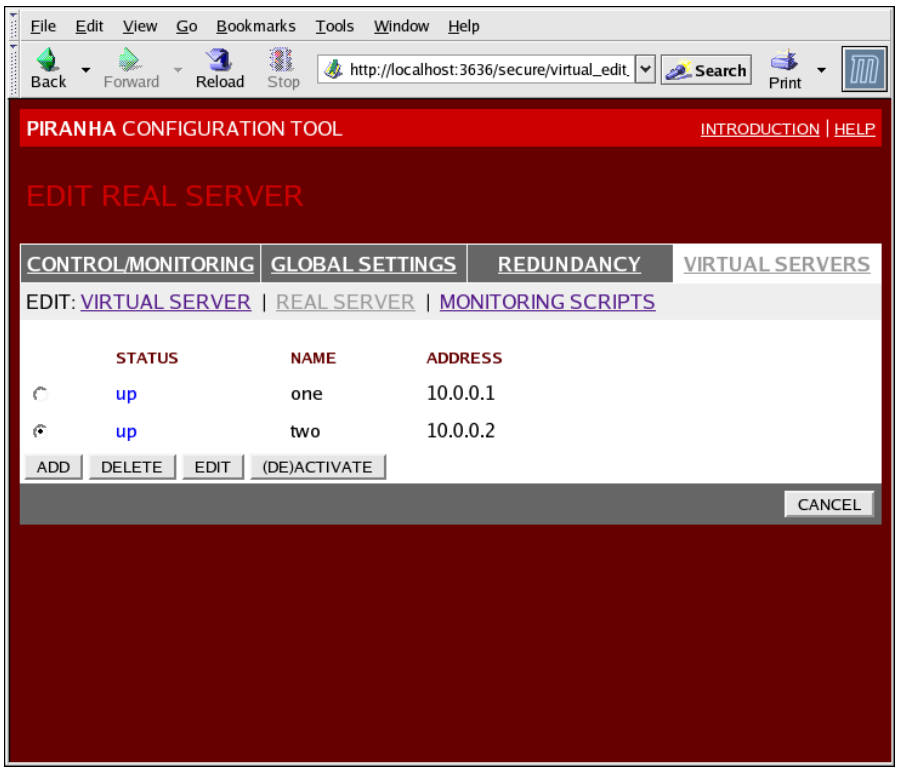

Figure 1.36. The REAL SERVER Subsection

Click the **ADD** button to add a new server. To delete an existing server, select the radio button beside it and click the **DELETE** button. Click the **EDIT** button to load the **EDIT REAL SERVER** panel, as seen in Figure 1.37, "The **REAL SERVER** Configuration Panel".

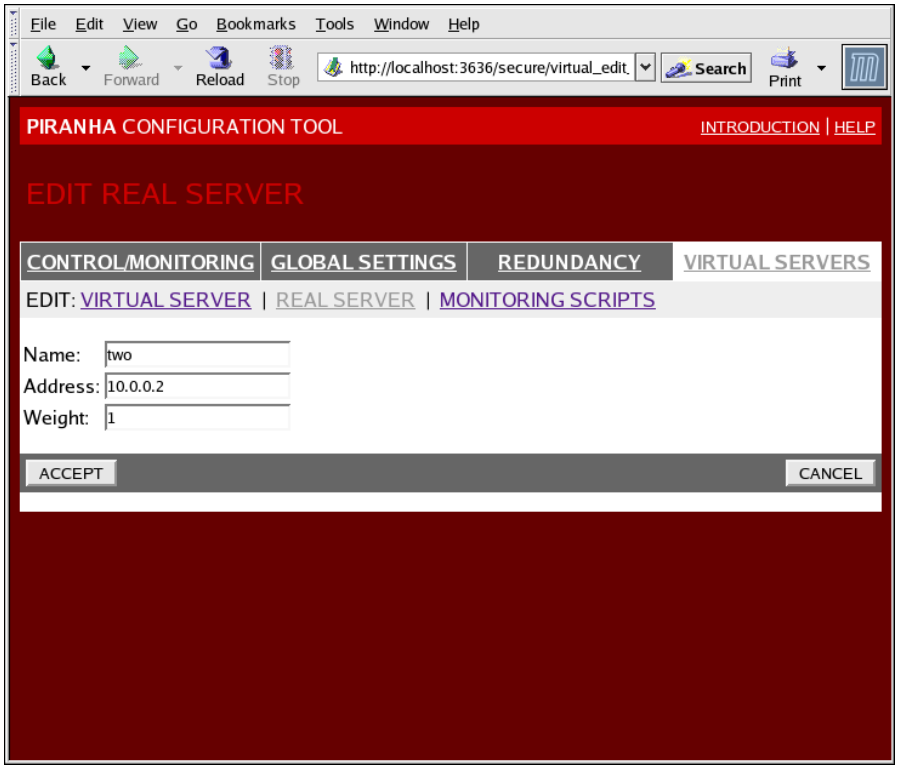

Figure 1.37. The REAL SERVER Configuration Panel

This panel consists of three entry fields:

Name

A descriptive name for the real server.

Note

This name is *not* the hostname for the machine, so make it descriptive and easily identifiable.

Address

The real server's IP address. Since the listening port is already specified for the associated virtual server, do not add a port number.

Weight

An integer value indicating this host's capacity relative to that of other hosts in the pool. The value can be arbitrary, but treat it as a ratio in relation to other real servers.

1.10.4.3. EDIT MONITORING SCRIPTS Subsection

Click on the **MONITORING SCRIPTS** link at the top of the page. The **EDIT MONITORING SCRIPTS** subsection allows the administrator to specify a send/expect string sequence to verify that the service for the virtual server is functional on each real server. It is also the place where the administrator can specify customized scripts to check services requiring dynamically changing data.

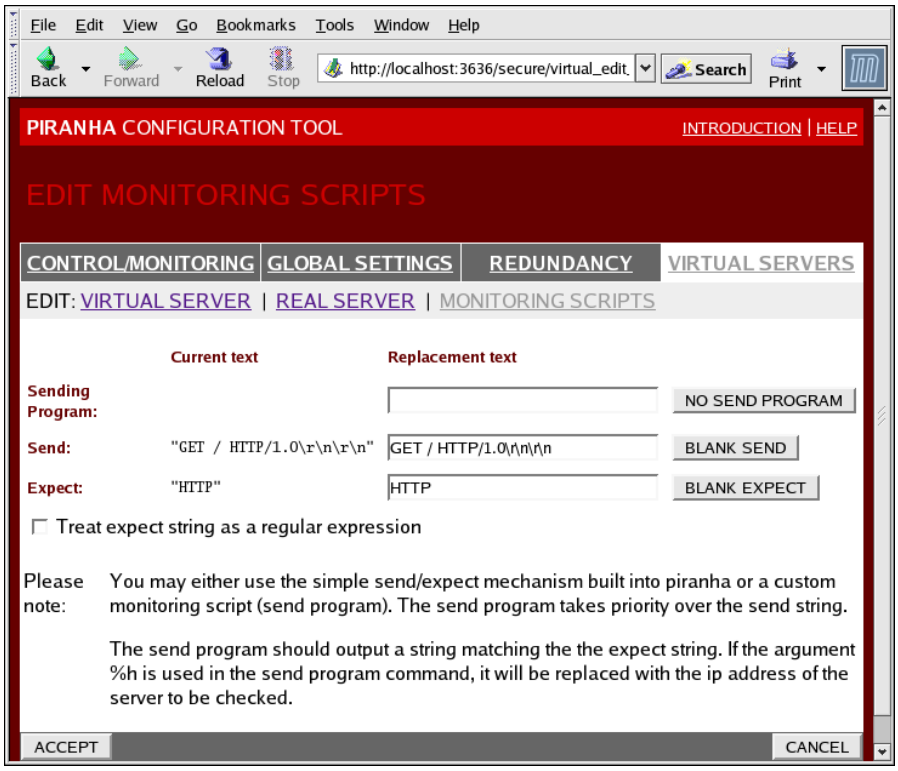
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_edit
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
EDIT MONITORING SCRIPTS
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
Current text
Replacement text
Sending Program:
NO SEND PROGRAM
Send:
"GET / HTTP/1.0\r\n\r\n"
GET / HTTP/1.0\r\n\r\n
BLANK SEND
Expect:
"HTTP"
HTTP
BLANK EXPECT
Treat expect string as a regular expression
Please note:
You may either use the simple send/expect mechanism built into piranha or a custom monitoring script (send program). The send program takes priority over the send string.
The send program should output a string matching the the expect string. If the argument %h is used in the send program command, it will be replaced with the ip address of the server to be checked.
ACCEPT
CANCEL

Figure 1.38. The EDIT MONITORING SCRIPTS Subsection

Sending Program

For more advanced service verification, you can use this field to specify the path to a service-checking script. This function is especially helpful for services that require dynamically changing data, such as HTTPS or SSL.

To use this function, you must write a script that returns a textual response, set it to be executable, and type the path to it in the **Sending Program** field.

Note

If an external program is entered in the **Sending Program** field, then the **Send** field is ignored.

Send

A string for the **nanny** daemon to send to each real server in this field. By default the send field is completed for HTTP. You can alter this value depending on your needs. If you leave this field blank, the **nanny** daemon attempts to open the port and assume the service is running if it succeeds.

Only one send sequence is allowed in this field, and it can only contain printable, ASCII characters as well as the following escape characters:

- \n for new line.
- \r for carriage return.
- \t for tab.
- \ to escape the next character which follows it.

Expect

The textual response the server should return if it is functioning properly. If you wrote your own sending program, enter the response you told it to send if it was successful.

[1] A virtual server is a service configured to listen on a specific virtual IP.

Chapter 2. Red Hat Cluster Suite Component Summary

This chapter provides a summary of Red Hat Cluster Suite components and consists of the following sections:

- Section 2.1, “Cluster Components”
- Section 2.2, “Man Pages”
- Section 2.3, “Compatible Hardware”

2.1. Cluster Components

Table 2.1, “Red Hat Cluster Manager Software Subsystem Components” summarizes Red Hat Cluster Suite components.

Table 2.1. Red Hat Cluster Manager Software Subsystem Components

Function	Components	Description
Conga	**luci**	Remote Management System - Management Station
	ricci	Remote Management System - Managed Station
Cluster Configuration Tool	**system-config-cluster**	Command used to manage cluster configuration in a graphical setting.
Cluster Logical Volume Manager (CLVM)	**clvmd**	The daemon that distributes LVM metadata updates around a cluster. It must be running on all nodes in the cluster and will give an error if a node in the cluster does not have this daemon running.
	lvm	LVM2 tools. Provides the command-line tools for LVM2..
	system-config-lvm	Provides graphical user interface for LVM2.
	lvm.conf	The LVM configuration file. The full path is **/etc/lvm/lvm.conf.**.
Cluster Configuration System (CCS)	**ccs_tool**	**ccs_tool** is part of the Cluster Configuration System (CCS). It is used to make online updates of CCS configuration files. Additionally, it can be used to upgrade cluster configuration files from CCS archives created with GFS 6.0 (and earlier) to the XML format configuration format used with this release of Red Hat Cluster Suite.
	ccs_test	Diagnostic and testing command that is used to retrieve information from configuration files through **ccsd**.
	ccsd	CCS daemon that runs on all cluster nodes and provides configuration file data to cluster software.
	cluster.conf	This is the cluster configuration file. The full path is **/etc/cluster/cluster.conf**.
Cluster Manager (CMAN)	**cman.ko**	The kernel module for CMAN.
	cman_tool	This is the administrative front end to CMAN. It starts and stops CMAN and can change some internal parameters such as votes.
	libcman.so.*<version number>*	Library for programs that need to interact with **cman.ko**.
Resource Group Manager (rgmanager)	**clusvcadm**	Command used to manually enable, disable, relocate, and restart user services in a cluster

	clustat	Command used to display the status of the cluster, including node membership and services running.
	clurgmgrd	Daemon used to handle user service requests including service start, service disable, service relocate, and service restart
	clurmtabd	Daemon used to handle Clustered NFS mount tables
Fence	**fence_apc**	Fence agent for APC power switch.
	fence_bladecenter	Fence agent for for IBM Bladecenters with Telnet interface.
	fence_bullpap	Fence agent for Bull Novascale Platform Administration Processor (PAP) Interface.
	fence_drac	Fencing agent for Dell Remote Access Card
	fence_ipmilan	Fence agent for machines controlled by IPMI (Intelligent Platform Management Interface) over LAN.
	fence_wti	Fence agent for WTI power switch.
	fence_brocade	Fence agent for Brocade Fibre Channel switch.
	fence_mcdata	Fence agent for McData Fibre Channel switch.
	fence_vixel	Fence agent for Vixel Fibre Channel switch.
	fence_sanbox2	Fence agent for SANBox2 Fibre Channel switch.
	fence_ilo	Fence agent for HP ILO interfaces (formerly fence_rib).
	fence_rsa	I/O Fencing agent for IBM RSA II.
	fence_gnbd	Fence agent used with GNBD storage.
	fence_scsi	I/O fencing agent for SCSI persistent reservations
	fence_egenera	Fence agent used with Egenera BladeFrame system.
	fence_manual	Fence agent for manual interaction. *NOTE* This component is not supported for production environments.
	fence_ack_manual	User interface for **fence_manual** agent.
	fence_node	A program which performs I/O fencing on a single node.
	fence_xvm	I/O Fencing agent for Xen virtual machines.
	fence_xvmd	I/O Fencing agent host for Xen virtual machines.
	fence_tool	A program to join and leave the fence

		domain.
	fenced	The I/O Fencing daemon.
DLM	**libdlm.so.<*version number*>**	Library for Distributed Lock Manager (DLM) support.
	dlm.ko	Kernel module that is installed on cluster nodes for Distributed Lock Manager (DLM) support.
GULM	**lock_gulmd**	Server/daemon that runs on each node and communicates with all nodes in GFS cluster.
	libgulm.so.*xxx*	Library for GULM lock manager support
	gulm_tool	Command that configures and debugs the **lock_gulmd** server.
GFS	**gfs.ko**	Kernel module that implements the GFS file system and is loaded on GFS cluster nodes.
	gfs_fsck	Command that repairs an unmounted GFS file system.
	gfs_grow	Command that grows a mounted GFS file system.
	gfs_jadd	Command that adds journals to a mounted GFS file system.
	gfs_mkfs	Command that creates a GFS file system on a storage device.
	gfs_quota	Command that manages quotas on a mounted GFS file system.
	gfs_tool	Command that configures or tunes a GFS file system. This command can also gather a variety of information about the file system.
	mount.gfs	Mount helper called by **mount(8)**; not used by user.
	lock_harness.ko	Implements a pluggable lock module interface for GFS that allows for a variety of locking mechanisms to be used.
	lock_dlm.ko	A lock module that implements DLM locking for GFS. It plugs into the lock harness, **lock_harness.ko** and communicates with the DLM lock manager in Red Hat Cluster Suite.
	lock_gulm.ko	A lock module that implements GULM locking for GFS. It plugs into the lock harness, **lock_harness.ko** and communicates with the GULM lock manager in Red Hat Cluster Suite.
	lock_nolock.ko	A lock module for use when GFS is used as a local file system only. It plugs into the lock harness, **lock_harness.ko** and provides local locking.

GNBD	**gnbd.ko**	Kernel module that implements the GNBD device driver on clients.
	gnbd_export	Command to create, export and manage GNBDs on a GNBD server.
	gnbd_import	Command to import and manage GNBDs on a GNBD client.
	gnbd_serv	A server daemon that allows a node to export local storage over the network.
LVS	**pulse**	This is the controlling process which starts all other daemons related to LVS routers. At boot time, the daemon is started by the **/etc/rc.d/init.d/pulse** script. It then reads the configuration file **/etc/sysconfig/ha/lvs.cf**. On the active LVS router, **pulse** starts the LVS daemon. On the backup router, **pulse** determines the health of the active router by executing a simple heartbeat at a user-configurable interval. If the active LVS router fails to respond after a user-configurable interval, it initiates failover. During failover, **pulse** on the backup LVS router instructs the **pulse** daemon on the active LVS router to shut down all LVS services, starts the **send_arp** program to reassign the floating IP addresses to the backup LVS router's MAC address, and starts the **lvs** daemon.
	lvsd	The **lvs** daemon runs on the active LVS router once called by **pulse**. It reads the configuration file **/etc/sysconfig/ha/lvs.cf**, calls the **ipvsadm** utility to build and maintain the IPVS routing table, and assigns a **nanny** process for each configured LVS service. If **nanny** reports a real server is down, **lvs** instructs the **ipvsadm** utility to remove the real server from the IPVS routing table.
	ipvsadm	This service updates the IPVS routing table in the kernel. The **lvs** daemon sets up and administers LVS by calling **ipvsadm** to add, change, or delete entries in the IPVS routing table.
	nanny	The **nanny** monitoring daemon runs on the active LVS router. Through this daemon, the active LVS router determines the health of each real server and, optionally, monitors its workload. A

		separate process runs for each service defined on each real server.
	lvs.cf	This is the LVS configuration file. The full path for the file is **/etc/sysconfig/ha/lvs.cf**. Directly or indirectly, all daemons get their configuration information from this file.
	Piranha Configuration Tool	This is the Web-based tool for monitoring, configuring, and administering LVS. This is the default tool to maintain the **/etc/sysconfig/ha/lvs.cf** LVS configuration file.
	send_arp	This program sends out ARP broadcasts when the floating IP address changes from one node to another during failover.
Quorum Disk	**qdisk**	A disk-based quorum daemon for CMAN / Linux-Cluster.
	mkqdisk	Cluster Quorum Disk Utility
	qdiskd	Cluster Quorum Disk Daemon

2.2. Man Pages

This section lists man pages that are relevant to Red Hat Cluster Suite, as an additional resource.

- Cluster Infrastructure
  - ccs_tool (8) - The tool used to make online updates of CCS config files
  - ccs_test (8) - The diagnostic tool for a running Cluster Configuration System
  - ccsd (8) - The daemon used to access CCS cluster configuration files
  - ccs (7) - Cluster Configuration System
  - cman_tool (8) - Cluster Management Tool
  - cluster.conf [cluster] (5) - The configuration file for cluster products
  - qdisk (5) - a disk-based quorum daemon for CMAN / Linux-Cluster
  - mkqdisk (8) - Cluster Quorum Disk Utility
  - qdiskd (8) - Cluster Quorum Disk Daemon
  - fence_ack_manual (8) - program run by an operator as a part of manual I/O Fencing
  - fence_apc (8) - I/O Fencing agent for APC MasterSwitch
  - fence_bladecenter (8) - I/O Fencing agent for IBM Bladecenter
  - fence_brocade (8) - I/O Fencing agent for Brocade FC switches
  - fence_bullpap (8) - I/O Fencing agent for Bull FAME architecture controlled by a PAP management console
  - fence_drac (8) - fencing agent for Dell Remote Access Card
  - fence_egenera (8) - I/O Fencing agent for the Egenera BladeFrame
  - fence_gnbd (8) - I/O Fencing agent for GNBD-based GFS clusters
  - fence_ilo (8) - I/O Fencing agent for HP Integrated Lights Out card
  - fence_ipmilan (8) - I/O Fencing agent for machines controlled by IPMI over LAN
  - fence_manual (8) - program run by fenced as a part of manual I/O Fencing

- fence_mcdata (8) - I/O Fencing agent for McData FC switches
- fence_node (8) - A program which performs I/O fencing on a single node
- fence_rib (8) - I/O Fencing agent for Compaq Remote Insight Lights Out card
- fence_rsa (8) - I/O Fencing agent for IBM RSA II
- fence_sanbox2 (8) - I/O Fencing agent for QLogic SANBox2 FC switches
- fence_scsi (8) - I/O fencing agent for SCSI persistent reservations
- fence_tool (8) - A program to join and leave the fence domain
- fence_vixel (8) - I/O Fencing agent for Vixel FC switches
- fence_wti (8) - I/O Fencing agent for WTI Network Power Switch
- fence_xvm (8) - I/O Fencing agent for Xen virtual machines
- fence_xvmd (8) - I/O Fencing agent host for Xen virtual machines
- fenced (8) - the I/O Fencing daemon

- High-availability Service Management
  - clusvcadm (8) - Cluster User Service Administration Utility
  - clustat (8) - Cluster Status Utility
  - Clurgmgrd [clurgmgrd] (8) - Resource Group (Cluster Service) Manager Daemon
  - clurmtabd (8) - Cluster NFS Remote Mount Table Daemon
- GFS
  - gfs_fsck (8) - Offline GFS file system checker
  - gfs_grow (8) - Expand a GFS filesystem
  - gfs_jadd (8) - Add journals to a GFS filesystem
  - gfs_mount (8) - GFS mount options
  - gfs_quota (8) - Manipulate GFS disk quotas
  - gfs_tool (8) - interface to gfs ioctl calls
- Cluster Logical Volume Manager
  - clvmd (8) - cluster LVM daemon
  - lvm (8) - LVM2 tools
  - lvm.conf [lvm] (5) - Configuration file for LVM2
  - lvmchange (8) - change attributes of the logical volume manager
  - pvcreate (8) - initialize a disk or partition for use by LVM
  - lvs (8) - report information about logical volumes
- Global Network Block Device
  - gnbd_export (8) - the interface to export GNBDs
  - gnbd_import (8) - manipulate GNBD block devices on a client
  - gnbd_serv (8) - gnbd server daemon
- LVS
  - pulse (8) - heartbeating daemon for monitoring the health of cluster nodes
  - lvs.cf [lvs] (5) - configuration file for lvs
  - lvscan (8) - scan (all disks) for logical volumes
  - lvsd (8) - daemon to control the Red Hat clustering services
  - ipvsadm (8) - Linux Virtual Server administration
  - ipvsadm-restore (8) - restore the IPVS table from stdin
  - ipvsadm-save (8) - save the IPVS table to stdout
  - nanny (8) - tool to monitor status of services in a cluster

- send_arp (8) - tool to notify network of a new IP address / MAC address mapping

2.3. Compatible Hardware

For information about hardware that is compatible with Red Hat Cluster Suite components (for example, supported fence devices, storage devices, and Fibre Channel switches), refer to the hardware configuration guidelines at http://www.redhat.com/cluster_suite/hardware/.

Revision History

Revision 1.0-6.400	**2013-10-31**	**Rüdiger Landmann**
Rebuild with publican 4.0.0		
Revision 1.0-6	**2012-07-18**	**Anthony Towns**
Rebuild for Publican 3.0		
Revision 1.0-0	**Fri Apr 24 2009**	

Index

A

about this document
- other Red Hat Enterprise Linux documents, Introduction

C

cluster
- displaying status, Cluster Status Tool

cluster administration
- displaying cluster and service status, Cluster Status Tool

cluster component compatible hardware, Compatible Hardware

cluster components table, Cluster Components

Cluster Configuration Tool
- accessing, Cluster Configuration Tool

cluster service
- displaying status, Cluster Status Tool

command line tools table, Command Line Administration Tools

compatible hardware
- cluster components, Compatible Hardware

Conga
- overview, Conga

Conga overview, Conga

F

feedback, Feedback

I

introduction, Introduction

L

LVS
- direct routing
  - requirements, hardware, Direct Routing
  - requirements, network, Direct Routing
  - requirements, software, Direct Routing
- routing methods
  - NAT, Routing Methods
- three tiered
  - high-availability cluster, Three-Tier LVS Topology

N

NAT
- routing methods, LVS, Routing Methods

network address translation (see NAT)

O

overview
- economy, Red Hat GFS
- performance, Red Hat GFS
- scalability, Red Hat GFS

P

Piranha Configuration Tool
- CONTROL/MONITORING, CONTROL/MONITORING
- EDIT MONITORING SCRIPTS Subsection, EDIT MONITORING SCRIPTS Subsection
- GLOBAL SETTINGS, GLOBAL SETTINGS
- login panel, Linux Virtual Server Administration GUI
- necessary software, Linux Virtual Server Administration GUI
- REAL SERVER subsection, REAL SERVER Subsection
- REDUNDANCY, REDUNDANCY
- VIRTUAL SERVER subsection, VIRTUAL SERVERS
  - Firewall Mark, The VIRTUAL SERVER Subsection
  - Persistence, The VIRTUAL SERVER Subsection
  - Scheduling, The VIRTUAL SERVER Subsection
  - Virtual IP Address, The VIRTUAL SERVER Subsection

- VIRTUAL SERVERS, VIRTUAL SERVERS

R

Red Hat Cluster Manager
- components, Cluster Components

T

table
- command line tools, Command Line Administration Tools

tables
- cluster components, Cluster Components